Mobile crowdsensing of parking space using geofencing and activity recognition

Mikko Rinne*, Seppo Törmä
Aalto University, PO Box 15400, FI-00076 Aalto, Finland, firstname.lastname@aalto.fi

Abstract
New mobile phones come with an increasing array of sensors. Recently mobile operating systems have started to incorporate means to offer contextual information derived from measurements of multiple sensors. A phone can be aware of whether it is transported in a vehicle or carried on foot. We investigated how these sensing capabilities could be used to derive information about available parking places in the absence of parking sensor infrastructure and tested the method with a dedicated mobile client connected to a network server. In this document we present the sensing method and the application, and qualitatively analyze the pros and cons of the approach.

Keywords: mobile, parking, sensor

1 Introduction

Finding a good place to park a car is a practical problem faced by millions of drivers daily. At personal level, it involves anxiety and uncertainty, and at the level of society, it wastes limited resources – time, road space, and fuel – when drivers circulate in search of free parking spaces. These problems could be alleviated if drivers had advance information of vacant parking spots. The information can be gathered with dedicated sensor systems keeping track of the reservation status of parking areas. However, installing network-connected sensors to all parking areas presents a considerable cost.

Information of vacant parking places can alternatively be gathered through *crowdsourcing* -- that is, as direct input from drivers. Crowdsourcing has previously been utilized in multiple parking assistance trials, e.g. [2, 4]. In the basic crowdsourcing approach drivers can assist each other by marking free parking spots against the benefit of receiving information from the community in return. *SpotScout*[1], founded in 2004, implemented a combination of parking

[1] http://www.bizjournals.com/boston/blog/mass-high-tech/2008/03/

garage infrastructure information, resale of private parking spaces and a market for on-street parking, where any parked driver could sell their departure time to the next one in need of a place. In 2010 Google Labs experimented with *OpenSpot*[2], with which drivers could mark the parking spots they vacate in crowded areas based on GPS locations. The system indicated age of the marking with colors and rewarded helpful drivers with Karma points. *ParkJam* [2] specializes on publishing the parking information as linked open data[3]. Until now systems with some infrastructure backing have been more successful than purely crowdsourced candidates. *Parkopedia*[4] uses crowdsourcing for rating locations and adding missing spaces but not to update real-time status. It currently (14.1.2014) lists over 28 million spaces in 40 countries. *ParkatmyHouse*[5] is a broker for privately owned parking space, where availability information is based on reservations booked through the system.

Earlier trials with crowdsourced parking space availability have not had the same array of mobile sensing technologies at their disposal. Combinations of sensors are starting to be used to provide contextual information about the activity taking place around the device. Information gathering from sensors of users' (mobile) devices is called *crowdsensing* [1]. In this paper we study, *whether smartphone sensor data could be utilized to automate the process of aiding drivers in finding vacant parking places*?

We approach the problem with a method utilizing primarily the activity recognition of two device states, *in vehicle* and *on foot*, combined with the location information available from satellite positioning. These simple events are used to derive the occurrence of a parking event, and consequently by means of complex event processing [3] to deduce that there was space to park in the designated area. The method is demonstrated by a parking assistant application running in Android OS and communicating with a network server.

The rest of the paper is organized as follows. In Section 2 we describe the algorithm used as basis for combined crowdsourced and crowdsensed detection. The new virtual sensors available in Android OS are presented in Section 3. In Section 4 we present the patterns of sensor events, which are used to detect parking. Our demonstration application is described in Section 5. Challenges of the approach and suggestions for potential solutions are collected to Section 6. Section 7 summarizes the concept and presents our conclusions, listing also potential future improvements.

---

spotscout-parks-100k-in-new-funding.html

[2] http://www.androidpolice.com/2010/07/09/
googles-open-spot-app-helps-you-find-a-parking-spot/

[3] http://linkeddata.org/

[4] http://parkopedia.com

[5] http://parkatmyhouse.com/

2 Crowdsensing parking place availability

If the success or failure of a parking attempt can be detected, other drivers can be informed about parking place availability. A successful attempt communicates that there was free space and therefore it is likely that free space is still available in the area (unless the previous driver occupied the last free slot). A failed attempt communicates that the parking area is full. In addition, when a driver releases a parking spot and drives away, it indicates that free space should be available again.

If detection is partial - that is, succeeded and failed attempts can be detected only from a subset of drivers, hereforth called *visible drivers* - the information can still be assumed valid, although only for a limited period of time. Since there are also *invisible drivers* (not contributing information) whose attempts cannot be detected, the information soon becomes obsolete, and the status of the parking area becomes unknown (Figure 1).

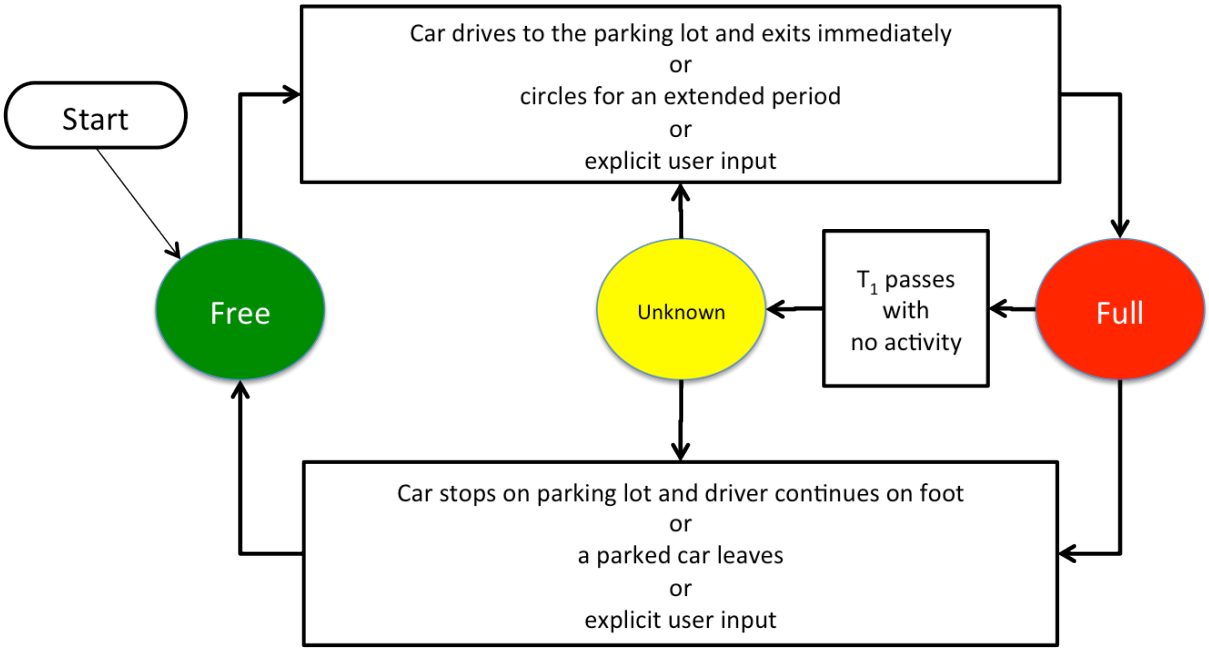

Figure 1: Algorithm to derive the status of a parking area

3 Sensors in Android

In the spring of 2013 Google introduced a set of new features in the location API of the Google Play Services[6]. Two particularly interesting additions from our target application point-of-view were:

[6] https://developer.android.com/google/play-services/location.html

- Geofencing API[7], which allows the specification of geographic areas (initially circular) and generates enter and exit events when a device crosses the border.
- Activity recognition API[8], which can provide updates of the activity type related to a device. The current types are
    - **still:** device is not moving
    - **tilting:** the position or orientation of the device is changing, indicating a change of activity type
    - **walking:** the user carrying the device is walking
    - **on bicycle:** the user is riding a bicycle
    - **in vehicle:** the user is in a vehicle

Both of these can be considered abstract sensors that combine and synthesize information from many hardware devices.

4 Patterns of succeeded and failed parking attempts

Based on the existence of geofencing and activity recognition the following event patterns can be identified:

1) Succeeded parking attempt at parking area r:
    a) enter r
    b) in vehicle
    c) on foot
    d) exit r
2) Failed parking attempt:
    a) enter r
    b) in vehicle
    c) exit r
3) Release of a parking space:
    a) enter r
    b) on foot
    c) in vehicle
    d) exit r

Reliable detection of the higher level events naturally requires that the underlying sensor platform provides the input events reliably. With reliable detection, one more level can be derived:

[7] https://developer.android.com/reference/com/google/android/gms/location/Geofence.html

[8] https://developer.android.com/reference/com/google/android/gms/location/ActivityRecognitionClient.html

1. Succeeded parking attempt => Area r had space
2. Failed parking attempt => Area r is full
3. Release of a parking space = >Area r has space

The reliability of the whole chain is tied to the reliability of the sensor input and the proportion of visible users from the total user base.

5 Demonstration application

A mobile Android client with accompanying server software has been created in research project *SPIRE*[9] (Smart Parking for Intelligent Real-Estate). Due to the importance of sensor input and especially activity recognition, the client was implemented as a native Android application. The architecture and main interactions between components are illustrated in Figure 2. The *mobile client* (1) uses Google cloud services[10] for sign-in, Google maps and destination searches. It sends input from the user and sensors (B) to our *HTTP server* (2) running over a *Hunchentoot*[11] Common Lisp framework. The HTTP server uses a *Sesame*[12] *RDF*[13] Graph Store (3) server for data storage, sending *SPARQL*[14] queries and update commands (D) and retrieving RDF-format data (E). For push-messaging to the Client (1), the HTTP server (2) sends notifications using *Google Cloud Messaging*[15]. The server can also retrieve input from infrastructure sensors (4), if available. In this case a policy for handling input from different sources is needed. If the infrastructure input is reliable, the primary policy could be not to collect end-user input for sensor-equipped parking lots at all. Another possibility is to collect end-user feedback for corrective actions in calibrating infrastructure sensor input. Loop sensors under driveways may get out of sync due to various reasons, and end-user input could be a fast way to re-calibrate the associated counters.

To test the application 138 parking areas from the Aalto University campus area were added to the database including e.g. location, radius, number of parking spaces and textual information. The mobile client user interface is shown in Figure 3. A map view of parking areas is shown in (a) together with the info-text for the selected parking area. The server provides the 20 largest parking areas in view for display. For each parking area the color of the symbol indicates the status (green, orange, red). If the status of any parking area changes,

---

9 http://www.hiit.fi/spire

10 http://developer.android.com/google/index.html

11 http://weitz.de/hunchentoot/

12 http://www.openrdf.org/

13 http://www.w3.org/RDF/

14 http://www.w3.org/TR/sparql11-query/

15 http://developer.android.com/google/gcm/index.html

the P-symbol color in the map view is automatically updated in all clients, where the symbol is displayed. Information on a sample parking lot is shown in (b). Any user can also manually input the status of a parking lot, as shown in (c). When the server detects parking, the client displays a silent notification to the user asking, whether the user would like to update status for the parking area. If no manual update is given, the system makes the default assumption that there was space and automatically marks the area green. Other client features include e.g. destination search, map window centering to current GPS location, favorite list management with up-to-date display of parking area status, one-click activation of Google Maps navigation to selected destination or parking area and provision of end-user surveys.

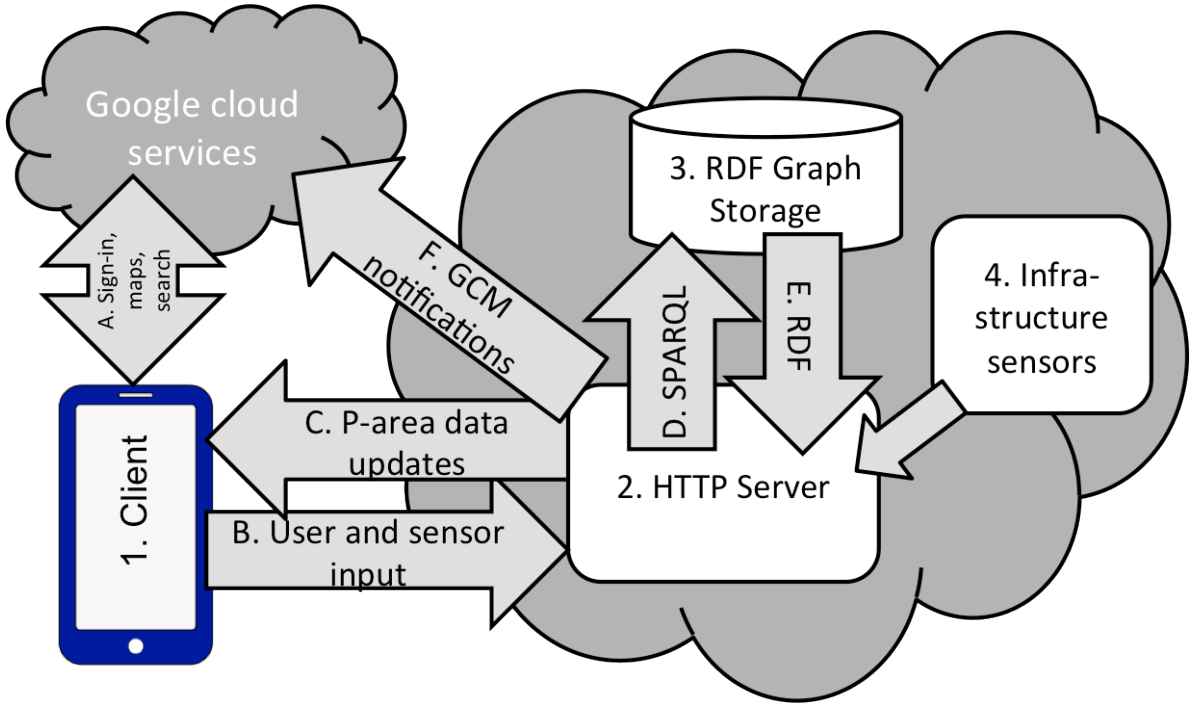

Figure 2: Parking software architecture and interactions

In this context we are more interested in what happens under the hood. The client activates a circular geofence around every parking lot displayed on the screen. The client reports the following events to our server:

- **Geofence crossings:** An entry/exit report with the corresponding parking area identification is sent every time a geofence is entered or exited, respectively.
- **Coordinates:** When inside a geofence, client location is sent with five second intervals.
- **Recognized activity:** When inside a geofence, every change between *in vehicle*, *on bicycle* and *on foot* is reported, starting from a zero-state.

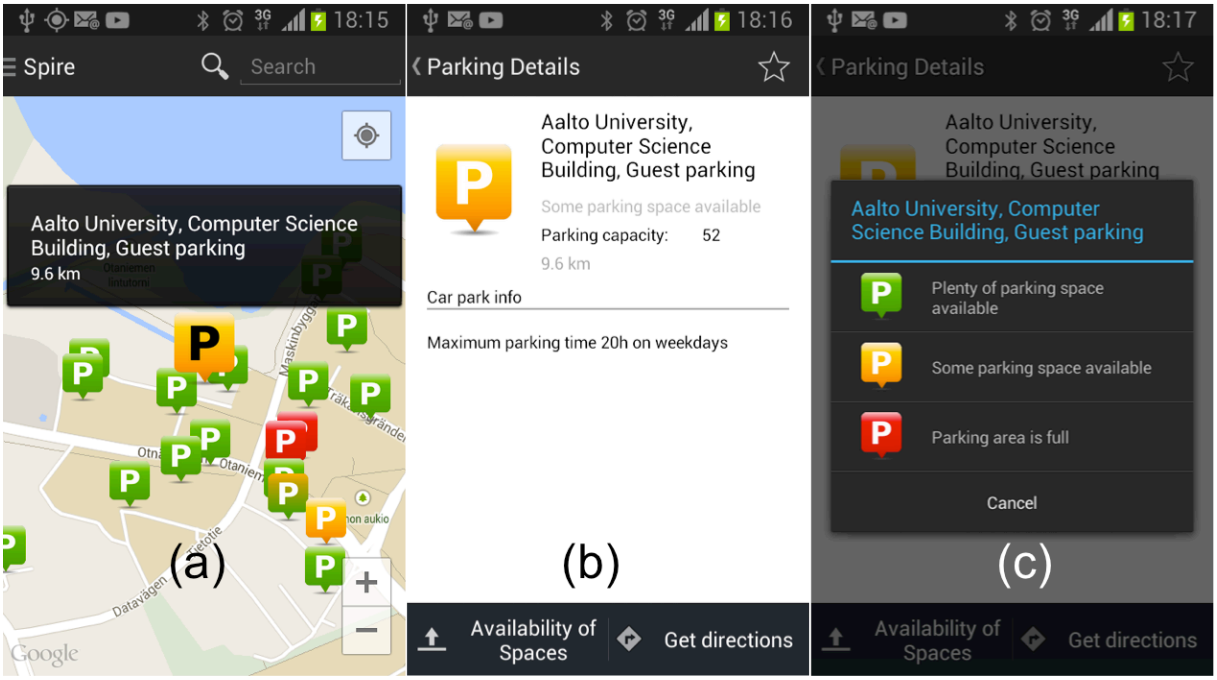

Figure 3: Mobile client screen captures of (a) map view, (b) parking lot info, (c) manual status input

An example log of a successfully detected parking procedure at the university campus is shown in Table 1. From this log it can be seen that the mobile client entered the parking lot in a vehicle and exited with a person travelling on foot. This sequence would result in the impacted parking area marked as having available space and a notification sent to the client signalling the detected parking and asking, whether the user would like to manually update the status of the parking area.

Table 1: Sample log of a successfully detected parking procedure

Time:	Type:	Event:
11:51:45	Geofence	<Computer Science Building, Guest parking> <Enter>
11:51:54	Activity	<IN_VEHICLE>
11:51:55	Location	60.187546 24.82136
	Location	...extra GPS coordinates removed...
11:52:40	Location	60.18752 24.821259
11:53:01	Activity	<ON_FOOT>
11:53:01	Geofence	<Computer Science Building, Guest parking> <Exit>

6 Challenges and potential solutions

There are a number of challenges and sources of imperfections in this approach. They can be roughly divided to three categories:

- Algorithmic imperfections

- Technical challenges
- Social issues

In the following each one of these will be discussed separately.

6.1 Algorithmic imperfections

Full parking area detection victim: One major drawback of the automatic detection is that it requires a victim. After a car gets the last available space, the area will still be marked as having free space. Only when a car needs to turn back without parking (or an active user manually marks the area as being full) the full-state is properly detected.

Drive-thru problem: Also in our test setup there are parking areas, which need to be driven through in order to access another parking area. Automatic separation of a simple drive-through from an unsuccessful parking attempt is very challenging. It could be attempted by an algorithm learning the driving habits of individual drivers and comparing driving parameters between *normal driving* and *driving while looking for a place to park*. We have not investigated this approach in detail, but expect it to be complicated and error-prone.

Invisible drivers: As discussed earlier, parking infrastructure sensors would be able to observe all vehicles, but our mobile client cannot be assumed to be running in all cars. Parking area status may change due to users invisible to the system and at least one visible user is needed to update the situation for others.

6.2 Technical challenges

Power consumption vs. geofence entry detection: The current Google location provider & geofence API only trigger a geofence entry when the full area of positioning uncertainty is contained inside the geofence. In practice this means that only satellite positioning is accurate enough to trigger geofence entry to a parking area. Forcing satellite positioning to run continuously has a high impact on power consumption in current devices and is therefore not feasible. After observing that power consumption for activity recognition was considerably lower than for satellite positioning, we settled on the compromise of continuously running activity recognition. If the device is observed to be moving longer than 15 seconds, satellite positioning is switched on. When the device becomes still again, satellite positioning is switched off. With this approach no major increase in power consumption was observed in connection with normal commuting and the geofence detections also worked when the device was locked or our service was running on the background.

Activity recognition uncertainty: Delayed, missed or wrong recognitions can significantly undermine the automatic detection functionality of the application. The activity recognition API provides the relative detection probabilities of all tested activities, allowing

adjustments between detection likelihood and reliability. So far our tests have mostly been suffering from missed detections, which would indicate that a lower reliability requirement might improve the situation.

6.3 Social issues

These challenges are related to the manual input provided by users, and are therefore common to all systems with crowdsourced parking availability.

Motivation: In order for end-users to be motivated to use the program and provide data, they should get something useful for themselves. First, there needs to be a problem: If parking space is easily available, there is no problem to be solved. Second, a sufficient number of visible users for the same parking lot are needed to produce a tangible benefit. This creates challenges in the startup phase of the system. The best scenario would be a group of people depending on the same scarce parking spaces deciding together to take the application into use. During startup phase gamification of the process could improve user motivation. In the case of paid parking, discounts could be granted based on frequent and truthful reports.

Individual benefit over common good: A user willing to secure a slot in his/her favorite parking area might be tempted to artificially mark an area full e.g. when leaving from home to divert other application users to other parking areas. Algorithmic detection of users, who constantly input data contradicting with others, could be added. Manual feedback from misbehaving users could be ignored or prioritized lower in setting the parking area status.

Malicious users: In any crowdsourced system there may be users, who are willing to confuse rather than to contribute. The right of any user to manually set the status of any parking area means that a malicious user can quickly input false information to multiple areas. This can be mitigated technically by restricting that only the status of nearby parking lots can be changed. Also the user profiling measures discussed in the context of individual benefit can be helpful.

7 Conclusions

More and more applications are being based on the ubiquitous availability of connected sensors in smartphones. In addition to the amount and penetration of smartphones, also the number of sensors per phone is increasing. Data derived from the sensors is getting more advanced, with activity recognition now being offered as a service by the operating system.

In this study we have considered a combined crowdsensing and crowdsourcing approach to parking. Users of a mobile application get a near-real-time view of the current parking situation on the map window of their choice. The application server can utilize any available information source to provide parking status information. The mobile application can collect end-user feedback on parking area status both through explicit feedback by the users and

through sensor-based estimation of the status of parking areas. At the time of writing the authors are not aware of other sources for automatic mobile sensor based parking availability detection.

The automatic sensor-based solution utilizes primarily two types of sensor inputs: geofences to indicate when a user enters or exits a parking area and activity recognition to tell, whether the user is currently moving by car or on foot. Sequences of these sensor observations can be used to derive whether the user (a) parked in the area, (b) was unsuccessfully looking for a place or (c) released a parking space. From these observations we can draw three higher-level conclusions:

- If the user was able to park, there was still space in the area (d)
- If a parking attempt was unsuccessful, the area is full (e)
- If a space was released, the area should no longer be full (f)

A clear benefit of the approach is the ability to assist drivers with real-time information on the current parking situation without any other supporting infrastructure than the cellular network and the smartphones of the car drivers. Based on both theory and experiences gained with the application, the following characteristics are beneficial for the crowdsensing approach:

- Reasonably large area size: Detection accuracy is improved, when both driving and walking in the parking area take more time. Also, if the status of a small parking area can be seen by drivers without entering, a full parking area cannot be automatically detected.
- Only one driving entrance: If there are multiple driveways, drivers may use the parking areas for drive-through and cause erroneous conclusions that the area was full.
- Isolated parking areas: Overlapping geofences of parking areas may cause confusion in detecting, which parking area was the real target. This will improve in the future when other geofence shapes (polygons) are introduced.
- Homogeneous parking permissions: Similar to the issue of parking area isolation, if one area contains multiple categories like openly available places, paid places, employee parking of a nearby company and some places reserved for the handicapped, the status of each category needs to be tracked individually. It may become very difficult to achieve both accurate detection and an adequate body of visible users per category.

The performance of the system can be improved by meeting as many of these conditions as possible. All improvements in activity recognition speed and accuracy improve overall performance and make it feasible to detect longer and more elaborate event patterns.

End-user motivation is a non-trivial issue. First, there needs to be a problem with scarcity before a driver can be motivated to seek assistance from a mobile application. Next, there has to be a tangible benefit, which in turn requires an adequate body of users. Attracting early

adopters will not be easy, because especially the first user doesn't get any benefit, unless the same application is usable also on parking areas equipped with infrastructure sensors. Finally, the application users are competitors for the same parking space. There needs to be a mindset of collaboration and common understanding that open sharing of information will be everyone's benefit. Problems due to misbehaving users can be mitigated by personal profile tracking and geographical restrictions to end-user input.

Some shortcomings are inherent to the approach and will always be present. A full parking lot cannot be detected in time by sensing user behaviour, only through manual reporting of the situation or by tracking a driver who cannot find a spot. Roadside parking, or very small parking areas where drivers can visually detect full occupation without driving into the area, cannot be detected by mobile sensing. If there is a need to drive through a parking area (e.g. drive past free 4h public parking places to reach full-day employee parking) it is very challenging to detect without explicit input, whether the driver wanted to park at all.

When these limitations are understood and good scenarios are found, crowdsensed parking assistance is a promising approach to help drivers help each other in places where infrastructure sensors are not available.

8 Acknowledgments

The work has been carried out in SPIRE and TrafficSense projects funded by Tekes and Aalto University. Mobile client programming by Citat Oy[16].

9 References

1. Ganti, R.K. et al.: Mobile Crowdsensing : Current State and Future Challenges. IEEE Commun. Mag. 49, November, 32–39 (2011).

2. Kopeck, J., Domingue, J.: ParkJam : crowdsourcing parking availability information with linked data (Demo). 9th Extended Semantic Web Conference (ESWC 2012). p. 5 , Heraklion, Crete, Greece (2012).

3. Luckham, D.: The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. Addison-Wesley Professional (2002).

4. Yan, T. et al.: CrowdPark : A Crowdsourcing-based Parking Reservation System for Mobile Phones. (2011).

[16] http://www.citat.fi/en/