parkITsmart: minimization of cruising for parking

Tsiaras, Christos ; Hobi, Livio ; Hofstetter, Fabian ; Liniger, Samuel ; Stiller, Burkhard

Abstract: Finding a parking space in urban areas is a daily challenge for drivers across the world, due to the increasing amount of vehicles and the limited amount of parking spaces. Drivers who are looking for a parking space in peak hours are often forced to drive around city blocks until they spot a free parking space. This process is termed in literature "cruising for parking" and is proven to (a) cost a lot of time and gas for drivers, (b) generate unnecessary traffic load, and (c) affect the environment negatively due to increased vehicle emissions. This work proposes a Parking Monitoring and Management System (PMMS) that collects, processes, and presents data about available parking spaces and their tariffs within a geographical region. The end-user application of the PMMS, parkITsmart, delivers at drivers bird's-eye view concerning the parking availability. To facilitate this, the PMMS gathers data from drivers', vehicles, their mobile phones, and Parking Inspectors (PIs). This work shows that in the Internet-of-Things (IoT) environment, "pairing" cars and drivers' mobile phones, collecting data from their sensors, and from PIs in a parking monitoring and management system, can decrease significantly cruising times for parking and can increase the time demands of the parking controlling process.

DOI: https://doi.org/10.1109/ICCCN.2015.7288448

parkITsmart: Minimization of Cruising for Parking

Christos Tsiaras, Livio Hobi, Fabian Hofstetter, Samuel Liniger, Burkhard Stiller
CSG@IfI, University of Zurich, Binzmühlestrasse 14, CH-8050 Zurich, Switzerland
Email: {tsiaras, stiller}@ifi.uzh.ch , {livio.hobi, fabian.hofstetter, samuel.liniger}@uzh.ch

***Abstract*—Finding a parking space in urban areas is a daily challenge for drivers across the world, due to the increasing amount of vehicles and the limited amount of parking spaces. Drivers who are looking for a parking space in peak hours are often forced to drive around city blocks until they spot a free parking space. This process is termed in literature "cruising for parking" and is proven to (a) cost a lot of time and gas for drivers, (b) generate unnecessary traffic load, and (c) affect the environment negatively due to increased vehicle emissions. This work proposes a Parking Monitoring and Management System (PMMS) that collects, processes, and presents data about available parking spaces and their tariffs within a geographical region. The end-user application of the PMMS, parkITsmart, delivers at drivers bird's-eye view concerning the parking availability. To facilitate this, the PMMS gathers data from drivers', vehicles, their mobile phones, and Parking Inspectors (PIs). This work shows that in the Internet-of-Things (IoT) environment, "pairing" cars and drivers' mobile phones, collecting data from their sensors, and from PIs in a parking monitoring and management system, can decrease significantly cruising times for parking and can increase the time demands of the parking controlling process.**

***Keywords*—Cruising, parking, congestion, mobile application, IoT, sensors, simulation**

I. Introduction

Finding a parking space in urban areas is a daily challenge for drivers across the world, due to the increasing amount of vehicles. Among many different sorts of congestion like the standard network flow congestion, there are also many forms of parking-related congestion [5]. This causes undesirable problems such as environmental issues, *e.g.*, (1) air pollution/noise increment, (2) energy consumption, and (3) parking space shortage [28]. Studies conducted between 1927 and 2001 in the center of major business districts in eleven cities on four continents came to the conclusion that up to 30% of vehicles in a traffic jam are looking for a parking space and the average time to find space available is eight minutes [30]. Given the increased number of cars since then, this problem can only be larger. Thus, the process of finding a parking space (a) costs a lot of time and gas for drivers, (b) generates an unnecessary traffic load, and (c) affects the environment negatively due to increased emissions.

Even a small amount of "cruising for parking" time per vehicle creates a surprising amount of vehicle traffic. *E.g.*, assume it takes eight minutes to find an available parking space and each parking space is occupied on average by 5 vehicles per day; each parking space generates 40 minutes of cruising for parking time per day. If the average speed while looking for a parking space is half of the maximum speed limit in a city (15 kilometers per hour in Zurich), then each parking space generates ten Vehicle Kilometers Traveled (VKT) per day. Thus, in a year, this cruising for parking results to 3'650 VKT per parking space. Since this cruising is added to the traffic that is already congested, it makes a bad situation even worse.

This work describes the process of designing and developing a prototype Parking Monitoring and Management System (PMMS), that collects and processes data about available parking places and their tariffs within a geographically well-defined area. The PMMS follows a crowdsourcing approach, where the system does require input from driver's smart phones, smart cars in the future, and from parking providers. The key feature of the PMMS is to deliver parking availability estimations on a geographical map, based on the data provided by drivers via a mobile application termed, parkITsmart. Other services, such as payment solutions, parking-time extension, parking location reminder, and parking expiration reminders are proposed to provide an incentive to drivers to use such an application and to contribute to the PMMS when parking availability estimation is targeted.

parkITsmart is a platform-independent mobile application developed in this work for the PMMS purposes. Thus, major mobile Operating Systems (OS), such as Android and iOS, are supported. Furthermore, parkITsmart can be used from drivers that are not familiar with parking-related regulations of an area. Therefore, parkITsmart is designed to handle multilingual interfaces concerning the User Interface (UI) and information about parking regulations for a given area. Finally, parkITsmart has Parking Inspector (PI) accounts that will serve parking monitoring purposes, such as parking controlling and fine issuing if necessary. Those interfaces can be used by authorities, such as the police or private companies that are responsible to monitor parking areas.

The motivation for this work is the challenge the Swiss city of Zurich faces to collect a reliable number of parking spaces within the city. In 2011 no up-to-date data were available [31]. However, the city faces problems with low availability of parking spaces and wants to raise the price for an effective utilization [32]. Nowadays, it is worth from a financial point of view to search for a parking space on the street instead of parking the vehicle in a carpark. Therefore, it is proposed by the city of Zurich to raise the price of parking spaces to the level of carparks to avoid cruising for parking. However, according to [30] this approach would have an effect only, if the price of parking spaces on the street would approach the price of carparks. Thus, a suitable approach towards the cruising for parking minimization in Zurich and possibly other cities is needed.

The use of smartphones has increased in the last few years [14], which opened many possibilities to assist humans in every day life. Especially contracts for broadband access on mobile devices increased significantly in the last five years in Switzerland [6]. To overcome the parking problem, new technologies such as smartphones interconnected with vehicles in the scope of Internet-ofThings (IoT), can be used to show parking availability and to guide drivers to free parking spaces. The increase in broadband access on mobile devices brings up new possibilities, such as displaying real-time parking data on mobile Web maps, such as Google Maps [13].

There has already been put a lot of effort in this topic to find possible solutions which solve the cruising for parking problem. However, either these approaches are very expensive due to investments in infrastructure [20],[26] or systems suffered from a lack of use [12] and, therefore, did not work because the data needed were not gathered. The key goal of this paper is the increment of existing space's utilization without a cost-intensive solution, while using state-of-the-art technologies. The PMMS presented here does not require sensors to be deployed on each parking space. If vehicles are marked as parked by parkITsmart users, unavailable parking spaces can be displayed. Integrating this information with the total number of parking spaces in a certain area generates parking spaces availability information that is available to drivers looking for a parking space. This helps to decrease the cruising for parking, because drivers most likely will avoid to look for a parking space in places where they do not exist. To overcome the problem of not having a sufficient amount of users to collect data, parking information is gathered automatically from drivers' smartphones using methods presented in [34]. However, information is also gathered from PIs, who control parked vehicles, and possibly in the future by smart cars.

The remainder of this paper is structured as follows. Related work is discussed in Section II, followed by the PMMS architecture in Section III. Results of this work are presented in Section IV. Section V presents future work and possible extensions of the PMMS. Finally, Section VI summarizes this paper and draws conclusions.

II. Related Work

Today's technologies make it possible to collect updated real-time parking information, such as parking location, capacity, parking fee, and current availability of different parking facilities, which can be used for a more intelligent parking guidance [28]. This can be summarized under the term "smart parking" that is considered to be a subgroup of "smart cities" [11]. There are two main paths to gather availability information of parking spaces: (a) Infrastructure-based solutions work with sensors on parking spaces and mobile phones to collect availability information and (b) crowdsourcing approaches, which gather information through the user-base of the system.

A. Infrastructure-based Solutions

There are many Parking Guidance System (PGS) approaches [1] which face the parking problem and contribute to smart parking. They vary in costs and also in necessary infrastructure. More infrastructure-based approaches are applied in: (1) Deutsche Telekom is running a project in the Italian city of Pisa together with Kiunsys [19], where sensors are placed on the floor of each parking space at the Piazza Carrara. These sensors detect whether spaces are free or occupied. The information is sent to the city's server over the mobile network and displayed on indication panels to guide drivers to free spaces. There exists also a mobile application guiding drivers to free spaces, which can also be used to pay for the parking space. (2) A similar approach with the use of sensors is followed by SFpark in San Francisco (SF), where parking spaces are equipped with occupancy sensors. Additionally, the "parking space market" in SF is regulated by demand-responsive pricing [26]. (3) ParkNet is following an infrastructure-based approach, where vehicles are equipped with ultrasonic sensors to monitor parking availability when driving by [20]. All infrastructure-based PGS solutions deliver accurate information. However, such solutions have the major drawback of the high infrastructure deployment and maintenance costs.

B. Crowdsourcing Approaches and Mobile Parking Platforms

A possible way to minimize the cost of gathering parking availability information is via crowdsourcing-based approaches. Drivers are contributing to the system by sharing empty or occupied parking space information. The more drivers use such a system, the more reliable the information it delivers. There are different solutions applying this approach, but none has been really successful by now. Google's open spot was launched in 2010 [12], but has not been successful due to insufficient user participation, which are needed to reach reliable data [27]. In Switzerland currently two mobile parking-related platforms exist. (a) ParkU and (b) parkit, showing parking spaces to drivers, following a crowdsourcing-based approach.

ParkU [25] is a solution to find or rent free and cheap parking spaces in the city. It consists of an online parking space market, where individuals and companies can rent on-demand private parking spaces for a fee. Available parking spaces are displayed on the Web site or on the ParkU smartphone app for iPhone and Android. The main features on the drivers' side encompass: (1) searching for parking spaces, (2) booking a parking space in advance or on-demand, (3) paying without coins or bills with the use of PayPal and credit cards, (4) archiving previous bookings, (5) adding booked spaces into the device calendar, (6) registering via Facebook or a separate ParkU account, (7) extension possibility of the parking duration if it is supported by the parking provider and (8) canceling possibility of future reservations. Parking providers can rent a parking space for a monetary compensation. The pricing and parking space availability schedule is decided on-demand by the parking provider.

Although ParkU is actively used by drivers, it suffers from several weaknesses. The major weakness is that only private spaces provided by companies and private persons are presented to drivers. ParkU follows a free market approach. Thus, the parking space owner can define the price of the parking space. This leads to large price diversion for parking spaces in the same area. Since the application provides information only about privately owned parking

spaces, end-users sometimes face difficulties to locate the space. Parking spaces presented in ParkU should be accessible without requiring access to a building, such that the parking space will be accessible by any potential driver wanting to reserve it. Although the owner of the space is renting the parking space, while assuming that the space is not illegally occupied, it might happen that there will be another vehicle illegally parked on the space once the user that had reserved the space will arrive in that specific location. Parking space owners can make profit by renting their property on-demand. However, another important drawback of ParkU is that there is no regulation, or taxation, for such income and this is the main reason why some cities want to ban it. Finally, there is no routing to the parking space integrated into the application, which makes its usage harder.

Another parking platform similar to ParkU is parkit [23]. In parkit private persons, or parking providers can register to offer parking spaces. These parking spaces can be reserved for time slots with a mobile application. End-users can filter the availability of parking spaces with address and price preferences. parkit provides a navigation feature from the driver's current position to the parking space.

C. Hybrid Solution

[34] presents a crowdsourcing approach where drivers use an application using sensors to detect when a vehicle is parked. The mobile application (PhonePark) suggests to use (1) GPS, (2) the accelerometer, (3) pairing connections between the driver's mobile phone and the car's on-board Bluetooth, and (4) information from pay-by-phone parking services, to identify when a car is parked. Although PhonePark proposes a novel solution, the application is not available on-line and there are no energy-consumption-related information provided supporting its usability in a real-life scenario. PhonePark is rather a mechanism that is running on the mobile phone, while the PMMS here proposes (1) a complete parking management and monitoring solution, and (2) an extension to the PhonePark car pairing mechanism with iBeacons [2], and/or On-board Diagnostic (OBD) modules [22]. Last but not least, the frequent parking availability information proposed in this paper here by the PIs, can improve PhonePark's historical availability profile construction algorithm.

D. The Delta of parkITsmart

On the one hand, estimating availability of parking spaces is equivalent to collect information about unavailable parking spaces. Thus, a parking availability estimation system should know about the arrival and departure times of a vehicle. Thereafter, parkITsmart in addition to PhonePark shows: (1) unavailable parking spaces, (2) possible available parking spaces, and (3) availability in carparks. On the other hand, ParkU and parkit represent a fragment of all parking spaces in a city, since they cover private parking spaces only. Therefore, ParkU and parkit can be considered as partial solutions to the cruising for parking problem, which still remains unsolved. In contrast, parkITsmart supports in addition to parking detection techniques as of [34] the aggregation of data originating from different sources, such as parking providers and vehicle drivers, and integrates them into a single system to provide to drivers an overall view of the parking situation in a given area. Thus, parkITsmart identifies available parking spaces on streets, collects unavailability on those spaces, and summarizes availability in carparks. The data collection is done following a crowdsourcing approach. To overcome the problem of not having a large end-user-base providing data and no data concerning unavailability, parkITsmart uses the input from parking providers who have an incentive to control parking. Parking spaces are inspected frequently by parking controllers. Therefore, the occupied spaces detected during the controlling process can be integrated and used to present the unavailability of parking spaces. parkITsmart does neither require any costly infrastructure, nor demands sensor deployment on parking spaces.

III. PMMS

Delivering reliable parking availability information allowing end-users to park their vehicles with minimum cost, while respecting local regulations, demands data availability in two domains. Firstly, a database of the location of available parking spaces is needed, containing several information concerning a parking space, such as (a) geographic coordinates, (b) parking space's surface, (c) orientation, (d) regulations, and (e) pricing. Secondly, the status of the availability of a parking space is needed. Such information can be retrieved (1) by drivers that park and use their mobile phones to pay the fee for parking, (2) by parking controllers inspecting parked vehicles, and (3) from any possibly available sensors. Driven by the high-level architecture's design of PMMS a formal definition of the chosen architecture and an overview of the architecture and its implemented system. Additionally, front-end components and frameworks used, the back-end architecture and database, and how different parts communicate with each other is discussed.

A. The PMMS Architecture

The PMMS uses a centralized client-server architecture consistsing of three layers. The (1) User Interface (UI), (2) end-users' and PIs' mobile applications, and (3) database layer. Clients act on the UI layer and request information from the application server. The application server provides services for the client and if necessary accesses the database layer to retrieve data requested.

PMMS consists of four different parts (cf. Figure 1). Two front-end mobile applications, the Web server, and the database. The front-end mobile applications access the database through the Web server, and provide access for users to the features of parkITsmart. Therefore, the front-end mobile applications provide the connection between the user and the back-end of the PMMS. There are two different user groups parkITsmart has to serve. First, the driver's group including end-users seeking for available parking spaces. Second, PIs being responsible to control parked vehicles in a city. To serve the needs of those two user groups, two mobile applications were designed and implemented. On one hand, the end-user's application for drivers' needs, and on the other hand, the controllers' application to satisfy requirements of PIs.

The evaluation of the technology to implement these front-end applications was divided into four steps: requirements elicitation, platform-framework selection,

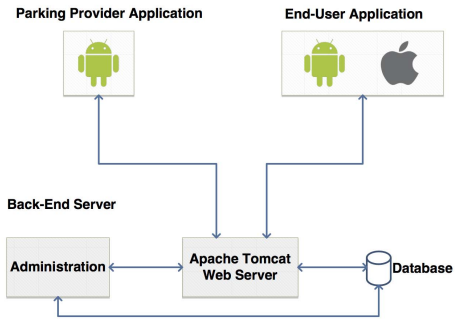

Figure 1. Overall Architecture of the PMMS

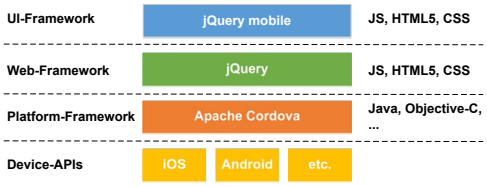

Figure 2. The Technology Stack of the Used Frameworks for parkITsmart

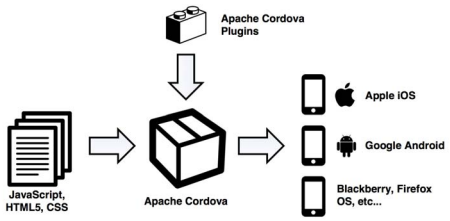

Figure 3. The Build Process of parkITsmart

UI-framework selection, and critical requirements testing. This procedure was chosen to ensure (a) reliable parking availability informations and (b) that the framework and platform chosen do not influence negatively the quality of parkITsmart. This four-step evaluation process leads to the combination of Apache Cordova, jQuery, and jQuery mobile to develop the front-end applications (cf. Figure 2). Apache Cordova [3] is a mobile development framework allowing access to device-specific functions and sensors with JavaScript code. Cordova supports Application Programming Interfaces (API) for major mobile platforms, such as iOS, Android, Windows mobile, and Blackberry. Thus, Cordova offers the possibility to build, update, and maintain applications with JavaScript, HyperText Markup Language five (HTML5), and Cascading Style Sheets (CSS) for multiple platforms with only one source code. The Cordova framework builds for every selected mobile platform a native application from the JavaScript, HTML5 and CSS source code (cf. Figure 3). The resulting application presents an internal Web view of the source code enriched with the functionality of device-specific APIs. This internal Web view is basically the device browser showing the Web page. However, the usage of the Web view is on some mobile platforms restricted and may lead to lower performance on demanding tasks compared to native applications [8]. To overcome this problem, jQuery [16] is a lightweight JavaScript library, which enables the developer to fast and easily traverse and manipulate HTML documents, supports JavaScript event handling, and simplifies Ajax communication. The library supports the vast majority of mobile as well as desktop browsers. The jQuery mobile [17] framework is a HTML5 user-interface framework to build responsive applications and Web sites mainly for mobile phones and tablets. It is based on the jQuery library and enables the developer to build responsive designs for multi-platform applications.

The back-end consists of two main components: the Web server and the database. The Web server provides services to the end-user applications and accesses the database when needed. The PMMS uses an Apache Tomcat [4] for Web services. Apache Tomcat is an open-source Web server, which supports Java Servlets [15]. Java Servlets provide a simple way to add Java-written Web services on top of the Tomcat server. The server runs two different Java Servlet applications, one for the controller and one for the driver's application. To store data, the system uses a MySQL database [21]. MySQL is a very popular, well supported, and easily usable open-source database management system.

The communication between the different components of the PMMS is a key feature of the entire system. The communication is designed to be fast, simple, secure, and reliable, to ensure fast communication and achieve good end-user experience. Furthermore, the driver's and the PI's application transmit sensitive user-related data, such as the address of a driver, vehicle details, such as license plates number, vehicle's parked position and details of payment instruments. Thus, the client-server communication uses the Secure Hypertext Transfer Protocol (HTTPS). Since the end-user applications are designed to be used on mobile devices and transfer data over mobile networks, the communication overhead must be small, because data communication costs battery and data transfer over Mobile Network Operators (MNO) might be costly. Thus, the communication between the different components is done by JavaScript Object Notation Remote Procedure Calls (JSON-RPC). JSON is a light-weight data-interchange format, where data is accessible as JSON objects in the JavaScript code [29]. The facts that this data format is very light-weight and also easy to process determined the reasons why JSON-RPC was chosen instead of any other possible communication protocol like XML-RPC [33]. There are different frameworks for JSON-RPC. Java and JavaScript are supported, due to the fact that the communication is based on Java Servlets and JavaScript. Thus, the framework of Ritwik Saikia [18] was chosen.

B. parkITsmart Usage

The collection of data concerning parked vehicles is achieved by two data sources. (1) Drivers can use the parkITsmart application (cf. Figure 4) to find a parking space. Once a driver decides where to park, the local regulations are fetched and according to the driver's profile the maximum amount of parking time is registered to the nearest parking space of the driver's location. A reminder with the parking expiration time and the vehicle's location is created. (2) An Near Field Communication (NFC) tag and a Quick Response (QR) code storing information concerning the vehicle are

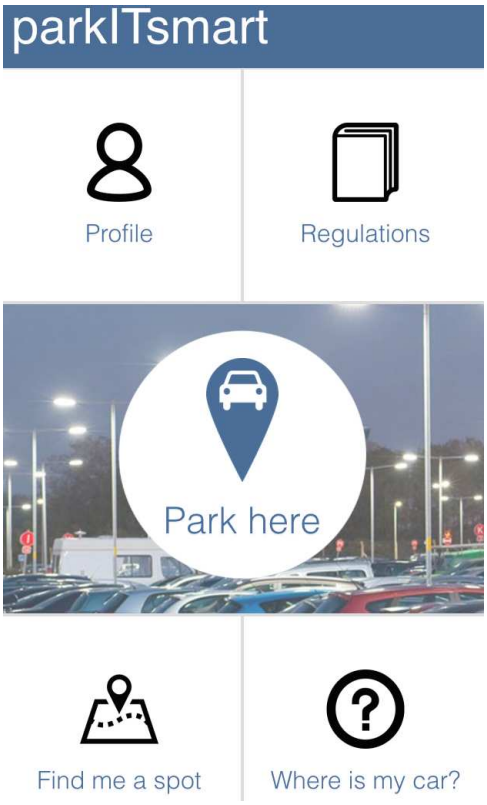

Figure 4. The parkITsmart Driver's UI [24].

integrated on the vehicle's parking permit that is visible from PIs. The PI scans each vehicle parked in an area with his mobile application and checks, if the vehicle is legally parked. The parking space closest to the parked vehicle is marked as occupied for the maximum time allowed according to regulations applicable at this location.

IV. Results

The PMMS was evaluated in a simulated environment in terms of (a) cruising for parking minimization and (b) parking controlling time efficiency improvement.

A. Cruising for Parking Minimization

An improvement of the cruising for parking minimization is beneficial for (a) the user who saves time, (b) the environment due to reduced vehicle emissions, (c) parking providers that increase the utilization of parking spaces, and (d) authorities, which achieve a lower traffic congestion rate. To model and evaluate the influence of parkITsmart in real life, this work compares a random routing time needed for a driver of a vehicle to reach a parking space, with the routing time needed when availability information is provided. It is assumed that a driver having no information about available parking spaces in a target area drives randomly until an available parking space is found. In contrast, a driver holding information about the closest available parking space will directly head to it.

The simulation models a part of a city using a grid, where every square of the grid represents one parking space, either available, or occupied. The squares of the grid correspond to a free parking space with a certain probability, which is parametrizable. Multiple vehicles start at random positions and head to random destinations. Vehicles move in a turn-based fashion from one square to an adjacent square. For simplicity reasons diagonal movements are not allowed. Two parameters model the total time (t_{tot}) needed to reach a destination (parking space). Equation 1 calculates the total duration by adding the time needed by a vehicle (t_{veh}) and on foot (t_{foot}). In this realistic simulation two vehicles heading to the same free space results in the first arriving vehicle is able to park there, the second vehicle has to look for another parking space.

$$t_{tot} = t_{veh} + t_{foot} \tag{1}$$

Table I. Scenario 1: One Square Kilometer in the City Center of Zurich at Noon

Parameter	Value
Vehicle speed	30 km/h
Percentage of available parking spaces	20%
Number of total parking spaces	100
Number of vehicles	10

Table II. Scenario 2: One Square Kilometer on a City's Suburb at 10:00 o'clock in the Morning

Parameter	Value
Vehicle speed	50 km/h
Percentage of available parking spaces	70%
Number of total parking spaces	100
Number of vehicles	5

Two scenarios have been simulated to examine the cruising for parking total time with and without parking availability information: (1) Cruising for parking in the city center and (2) cruising for parking in the city's suburb. Both scenarios share two common parameters, the grid size and the duration it takes to move from one location to a neighboring location on foot. However, these scenarios differ in the three following parameters:

- The average speed of vehicles
- The percentage of available parking spaces
- The amount of vehicles seeking for a parking space

Scenario (1) represents a typical situation in a city center, where vehicles are driven slowly, only a few parking spaces are available, and the amount of vehicles looking for a parking space is high. Table I shows the scenario of one square kilometer in the city center of Zurich at noon. These parameters are selected based on assumptions and information gained from the analysis of the historic data concerning carparks' availability of Zurich.

In contrast, scenario (2) covers a situation outside the city center in the morning, where vehicles may drive faster, the amount of free spaces is larger compared to (1) and the number of vehicles seeking for a parking space is lower than (1). Table II shows those parameters for the second scenario in the city's suburb at ten o'clock in the morning.

To increase the confidence of results the simulation of each scenario was executed 100 times. Over all iterations the average time to reach destinations was computed. Every iteration uses a new random grid, where every square has a parking space available based on the probability of available

parking spaces. The starting and destination positions are also selected randomly. Within each scenario parameters as stated in Table I and II were the same for every iteration.

For the simulation the following general and scenario-specific assumptions were made.

1) General Assumptions:

- The size of the grid is 10x10 squares, where one square equals 100 meters.
- The speed of vehicles is higher outside of the city center.
- The percentage of available parking spaces is lower in the city center.
- There are more vehicles looking for a parking space in the city center.
- The duration to walk to an adjacent square is 1.5 time units (assuming that a person walks 4 km/h by foot or 1.5 minutes for 100 meters).

2) Random Routing Assumptions:

- Vehicles are driving randomly without any information.
- If an available parking space is found the vehicle is parked and the remaining distance is walked.

3) Routing with Information Assumptions:

- The vehicle drives directly to the closest space to the destination and the driver walks the remaining distance.
- If the closest space to the goal of a driver gets occupied upon the approach, the vehicle heads to the next closest one.

Figure 5 illustrates an example instantiation of the simulation with two vehicles, where the vehicle at *Start 1* heads to destination 1 (*Dest. 1*), the vehicle at *Start 2* heads to destination 2 (*Dest. 2*) and both know where there are free spaces. Initially, both vehicles want to park at the available parking space between the two destinations, since it is the nearest one for both of them. After five moves vehicle two is at position 1 (*Pos. 1*) and vehicle one has parked at the nearest position. Vehicle two realizes this and, therefore, looks again for the closest parking space to its destination. This example shows how multiple vehicles may influence the average time of all vehicles to reach their destinations. Thus, the simulation's assumptions are close to the reality.

The results of this simulation can be seen in Table III. The ratio determines the benefit on cruising for parking between the approach with and without parking availability information. It is shown that cruising for parking with information about the availability of parking spaces is around 3.5 to 7.7 times lower than random cruising. This shows that the initial assumption that parking availability information reduces cruising for parking is valid. In the random approach each vehicle parks at the first free space and the driver walks for the rest of the distance. In the approach with parking availability information each vehicle seeks for the closest available parking space and parks there, if it is still available at the time of arrival. In the city center scenario the difference between the two approaches

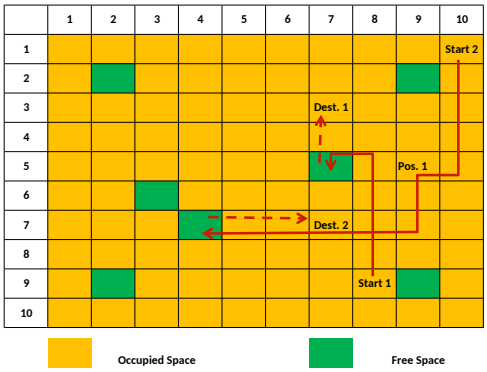

Figure 5. Example Simulation with Two Vehicles

Table III. THE RESULT OF THE SIMULATION

Scenario	Mean Time Units Random	Mean Time Units With Availability Information	Ratio
City center	11.12	3.2	3.48
Suburb	9.89	1.29	7.67

is not as high as in the second scenario one, since the number of free spaces is relatively small.

The PMMS has shown to be able to save time, fuel, and minimize vehicle emissions using information about availability of parking spaces. Simulation (1) and (2) results show a remarkable difference between the two approaches. However, this is not very surprising considering that a random routing is compared to a routing with parking availability information. The random routing assumes that drivers park on the first free parking space they encounter. This is, however, not very realistic, if they know that there are a lot of free spaces closer to the goal (like in scenario two). The result even in this simplified simulation shows that there are many opportunities to decrease cruising for parking using information about the availability of parking spaces.

B. Control Parking Efficiency Improvement

The second goal of parkITsmart, to increase the productivity of a Parking Inspector (PI), thus, needing less time for a PI to control a vehicle leads to the more vehicles being controlled in the time interval. This results in a revenue increase for the city, either because more illegally parked vehicles can be spotted or because fewer PIs are needed to perform the same task.

An estimation of the parking controlling process with the use of parkITsmart controller application leads to the following scenario (3) to be assumed. A PI in Zurich, Switzerland, controls a 100 meter long part of a street, which is occupied by vehicles. The average parking space length in Zurich is measured to be five meters long, and it is assumed that there is a total number of $N = 20$ vehicles that need to be controlled. The PI walks with an average speed of 4 km/h and it will cost $t_w = 1.5$ minutes to walk these 100 meters. Checking whether a vehicle is legally parked with the PIs application takes approximately $t_c = 3$ s, while using NFC tags or QR-codes. An ordinary check would demand the PI to check manually the arrival time of a vehicle by reading the time on a clock that marks the vehicle's arrival time in a location. This process approximately takes 5 s. If 10% of vehicles are parked illegally,

there are $I = 2$ illegally parked vehicles. Sending a fine with the parkITsmart controller applications has been measured to take at most $t_f = 30$ s, while it is observed that a PI in Zurich takes at most two minutes to register a fine without the parkITsmart application. Thus, the total vehicle controlling time t_{total} is $t_{total} = t_w + (t_s \cdot N) + (I \cdot t_f)$.

The time used to control 20 vehicles with 10% illegally parked vehicles rate without the PIs application:

$$t_{total}^{WithoutApplication} = 90\ s + (5\ s \cdot 20) + (120\ s \cdot 2) = 430\ s$$

The time used to control 20 vehicles with 10% illegally parked vehicles rate with the PIs application is:

$$t_{total}^{WithApplication} = 90\ s + (3\ s \cdot 20) + (30\ s \cdot 2) = 210\ s$$

This estimation shows that using the parkITsmart controlling application saves half of the controlling time. Walking takes a crucial amount of time, which has not been improved yet. Once there is a sufficient amount of data in the PMMS, a more sophisticated routing of PIs will save further time. Additionally, errors due to the human factor, such as wrong noting of license plates or issuing a fine to a vehicle, which is legally parked, will be minimized with parkITsmart. Eventually, this will increase the credibility of the parking controlling process.

V. Future Work

Sensors placed on the ground, can detect accurately the position of parked vehicles [1]. However, such solutions are costly to deploy and maintain. Furthermore, to minimize cruising for parking, it is shown that the total number of the parked vehicles on the streets, if also the number of available parking spaces is known, is sufficient. The precise location of a parking space is not mandatory to find an available space on a location (*e.g.*, a street or a block). Thus, this work extends the sensor-less approach by introducing a solution to count parked vehicles with the use of a mobile application. In the future, the accuracy of parkITsmart will be improved by bluetooth beacons (*e.g.*, iBeacons) or smart OBD connectors (*e.g.*, Freematics [10]) that will be placed in vehicles and paired with drivers' mobile phones. However, any placement of radio transmitting devices raises privacy-related issues, since driver's historic location data could be collected. Thus, the transmitting radius of such devices has to be selected to be small to cover only a small area around the vehicle.

Figure 6 shows a parking area next to the University of Zurich premises in Oerlikon with approximately 108 parking spaces in total. Figure 7 shows the same location at a different time, when approximately 56 vehicles are parked. On one hand, for a driver looking for a parking space in one area, a precise information of where each parked vehicle is parked is not needed. On the other hand, the number of remaining spaces $(108-56 = 52)$ is a useful information. iBeacons have (a) low cost and (b) low power demands. Assuming that each vehicle in the future will be equipped with an iBeacon transmitter, which transmits only when the vehicle is parked, the total number of vehicles can be collected by a moving iBeacon receiver, such as a parking controller's device or other moving vehicles. Deducting the number of parked vehicles from the total number of parking spaces available in an area can deliver

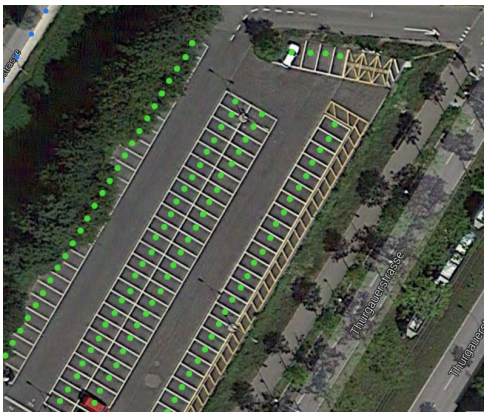

Figure 6. Parking Area with Approximately 108 Unoccupied Parking Spaces [13].

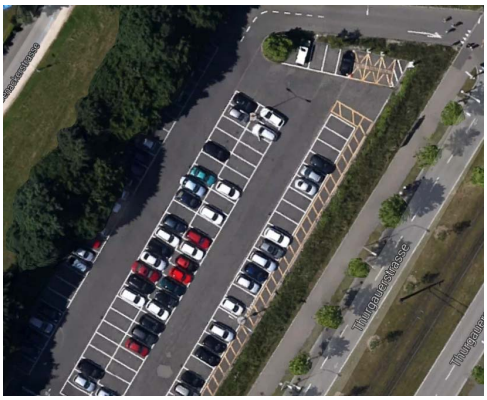

Figure 7. Parking Area with Approximately 56 Occupied Parking Spaces [13].

an accurate estimation of unoccupied parking spaces in that given area. The more frequent this information is updated, the closer to the real-time situation this information will be. Provided that the location of the iBeacon receiver is known, it is possible to use triangulation techniques [7][9] to estimate the location of parked vehicles.

Finally, the usability of parkITsmart can be improved in a next iteration, by choosing a native implementation for each platform. Moreover, a standardized approach to add additional parking regulations from additional cities and communities will increase the end-user-base. Last but not least, adding additional data concerning parking spaces in Zurich and other cities will improve significantly the parking space availability estimation.

VI. Summary and Conclusion

The outcome of this work is a prototype PMMS. The key feature is the delivery of a bird's-eye view of parking availability on a map, based on the data gathered by drivers, their smart-phones' and PIs. The prototype PMMS developed consists of two front-end applications and a back-end system. The drivers' application (parkITsmart) is publicly available in the Google Play Store for Android and in the Apple AppStore for iOS for the Swiss market [24]. The multilingual end-user application shows unavailable parking spaces on a

map, preesents local parking regulations, and lets drivers mark their vehicles as parked. The controller application is able to check, if a vehicle is parked legally by scanning an NFC-tag, QR-code, or by entering the vehicle's license plate manually. If the vehicle is illegally parked, the PI may issue a fine to the registered holder of the vehicle, which can be sent automatically to the driver by email, including multimedia files (such as a picture) proving the parking regulation violation.

The development of the parkITsmart prototype had to overcome a scalability-related challenge. The limited performance of the Cordova framework upon handling a lot of data points on a map requires several iterations and optimization steps to enable an acceptable user experience when searching for a parking space. The key to speed up the map functionality was to present clusters of parking spaces and not each parking space separately.

A simulation is used to assess the system developed and evaluate the primary goal of it to minimize cruising for parking. Results of this simulation showed that cruising for parking with such a system estimating parking space availability means approximately 3.5 to 7.7 times less cruising for parking to final destination time than without such information. With respect to the increment of PI's productivity and based on reasonable assumptions the PI with the controlling application checks the same amount of vehicles in about approximately 50% of the time of the entire vehicle controlling process. Thus, the PMMS shows that the key stakeholders, drivers, authorities, and parking providers, benefit in multiple domains.

ACKNOWLEDGMENT

This work was supported partially by the FLAMINGO project, funded by the EU FP7 Program under Contract No. FP7-2012-ICT-318488. Special thanks to our friends and colleagues Bénédict Birrer, Julia Breddermann, Mark Fishel, Elisabeth Hammer, Sarina Hobi, Lisa Kristiana, Tomas Ludrovan, Vanessa de Azevedo Machado, Guilherme Sperb Machado, Aleksei Mazhelis, Rania Nikolopoulou, Mirja Pulkkinen, Nicoletta Salmeri, Marina Josipovic, Andrei Vancea, Marja Weisskopf, Michael Weisskopf, and Chen Zhongheng for helping us to translate the parkITsmart application in 14 languages. Last but not least, special thanks are addressed to our colleagues Andri Lareida and Thomas Bocek for their priceless help during the testing period.

REFERENCES

[1] D. Burgstahler, F. Knapp, S. Zoller, T. Ruckelt, R. Steinmetz, *Where is that Car Parked? A Wireless Sensor Network-based Approach to Detect Car Positions*, IEEE 39th Conference on Local Computer Networks Workshops (LCN Workshops), pp. 514-522, Edmonton, Canada, September 2014.

[2] A. Cavalini, *iBeacons Bible 1.0*, URL: http://tiny.uzh.ch/dx, Visited in November 2014.

[3] Apache Cordova, URL: https://cordova.apache.org, Visited in September 2014.

[4] Apache Tomcat, URL: http://tomcat.apache.org/, Visited in October 2014.

[5] R. Arnott, T. Rave, R. Schöb, *Alleviating Urban Traffic Congestion*, MIT Press Books, 2005.

[6] Bundesamt für Kommunikation BAKOM, *Amtliche Fernmeldestatistik 2012*, URL: http://tiny.uzh.ch/dt, Visited in September 2014.

[7] J. Castano, M. Svensson, M. Ekstrom, *Local positioning for wireless sensors based on Bluetooth™* Radio and Wireless Conference, 2004 IEEE , pp. 195-198, 19-22 September 2004.

[8] S. Diwakar, *Titanium vs Phonegap vs Native application development*, URL: http://tiny.uzh.ch/du, Visited in October 2014

[9] S. Feldmann, T. Hartmann, K. Kyamakya, *An indoor Bluetooth-based positioning system: concept, implementation and experimental evaluation*, ICWN'03, Las Vegas, USA, 2003.

[10] Freematics, URL: http://freematics.com/, Visited in April 2015.

[11] T. Giuffrè, S.M. Siniscalchi, G. Tesoriere, *A novel architecture of parking management for smart cities.* Procedia - Soc. Behav. Sci. 53, 16-28, 2012

[12] A. Goodwin, *Open Spot app helps Android users find parking*, URL: http://tiny.uzh.ch/dv, Visited in October 2014.

[13] Google Maps, URL: http://www.maps.google.com, Visited in September 2014.

[14] IDC, *Worldwide Mobile Phone Market Forecast to Grow 7.3% in 2013 Driven by 1 Billion Smartphone Shipments*, URL: http://www.idc.com/getdoc.jsp?containerId=prUS24302813, Visited in September 2014.

[15] Java Servlets, URL: http://tiny.uzh.ch/dy, Visited in October 2014.

[16] jQuery, URL: http://jquery.com/, Visited in October 2014.

[17] jQuery mobile, URL: http://jquerymobile.com/, Visited in October 2014.

[18] JsonRpc, URL: https://github.com/RitwikSaikia/jsonrpc, Visited in October 2014.

[19] Kiunsys, URL: http://www.kiunsys.com/, Visited in November 2014.

[20] S. Mathur, T. Jin, N. Kasturirangan, J. Chandrasekaran, W. Xue, M. Gruteser, W. Trappe, *Parknet: Drive-by Sensing of Road-side Parking Statistics*, ACM MobiSys, 2010.

[21] MySQL Database, URL: http://www.mysql.com/, Visited in October 2014.

[22] OBD, URL: http://www.obdii.com/, Visited in April 2014.

[23] Park it, URL: http://www.parkit.ch, Visited in September 2014.

[24] ParkITsmart Webpage, URL: http://parkitsmart.com/, Visited in October 2014.

[25] Parku, URL: http://www.parku.ch, Visited in September 2014.

[26] SFpark, URL: http://sfpark.org, Visited in October 2014.

[27] I. Sherwin, *Google Labs' Open Spot: A Useful Application That No One Uses*, URL: http://tiny.uzh.ch/dw, Visited in October 2014.

[28] J. Shin, H. Jun, *A Study on Smart Parking Guidance Algorithm*, Transportation Research Part C: Emerging Technologies, 2014.

[29] S. Shin, Introduction to JSON (JavaScript Object Notation), URL: http://www.cse.iitd.ac.in/ cs5090250/JSON.pdf, Visited in October 2014.

[30] D. Shoup, *Cruising for Parking*, Transport Policy, Vol. 13, no. 3, 2006.

[31] Stadt Zürich Tiefbau- und Entsorgungsdepartement, *Wirtschaftliche Bedeutung von Parkplätzen in der Stadt Zürich*, Press Conference of 13. May 2011, URL: http://tiny.uzh.ch/ds, Visited in November 2014.

[32] Weisung des Stadtrats von Zürich an den Gemeinderat, *Motion von Gian von Planta und Markus Knauss betreffend Strassenparkplätze in der Innenstadt, Preiserhöhung für eine lenkungswirksame und effiziente Nutzung*, URL: http://tiny.uzh.ch/dr, Visited in November 2014.

[33] XML-RPC, URL: http://tiny.uzh.ch/dz, Visited in October 2014.

[34] B. Xu, O. Wolfson, J. Yang, L. Stenneth, P.S. Yu, P.C. Nelson, *Real-Time Street Parking Availability Estimation*, 14th International Conference on Mobile Data Management (MDM), Vol.1, pp.16-25, 3-6 June 2013.