Task-Aware Location-Based Services for Mobile Environments

FP7-SME-207-1-222292-TALOS

Content Aggregation D1.1

Deliverable lead contractor: IMIS – R.C. ATHENA

Panagiotis Georgantas pgeor@imis.athena-innovation.gr
Dieter Pfoser pfoser@cti.gr

Due data: 30.9.2009
Actual submission date: 13.11.2009

Abstract
This report summarizes the results of the Work Package "Context Aggregation" (WP1) of the TALOS project. A context aggregator has been implemented that can capture, analyze and aggregate the user's contextual attributes such as space and time.

Table of Contents

1 INTRODUCTION 5

2 OBJECTIVES CHANGE 5

3 EVALUATION OF APPROXIMATE POSITIONING TECHNIQUES 7

3.1 CENTROID 7

3.2 FINGERPRINTING 8

3.3 EVALUATION 9

3.4 CONCLUSIONS 12

4 ARCHITECTURE AND IMPLEMENTATION 13

4.1 ARCHITECTURE 13

4.2 IMPLEMENTATION DETAILS 14

5 COMPONENTS 16

5.1 APPROXIMATE POSITIONING SUPPORT TOOLS 16

5.2 CONTEXT AGGREGATOR 20

CONCLUSIONS 25

REFERENCES 26

APPENDIX 27

1 Introduction

This report summarizes the results of the Work Package "Context Aggregation" (WP1) of the TALOS project. A context aggregator has been implemented that can capture, analyze and aggregate the user's contextual attributes such as space and time. The implementation of a position estimation framework enables the user to estimate her position in a completely autonomous manner and without the requirement of any carrier or other third-party server intervention, both in outdoors and indoors environments. The results of this work are based on a wide and detailed analysis and evaluation of approximate positioning techniques that exploit standard GSM networks and WiFi hotspots. A suite of tools that support the operation of the position estimation framework has also been implemented.

The appendix includes the following scientific article that showcases and evaluations the use of soft positioning technology as a replacement for GPS when tracking vehicle in a road network. The article uses pre-existing map-matching technology developed by CTI for GPS tracking of vehicles and in this case applies to WiFi-based positioning.

Spiros Athanasiou, Panos Georgantas, George Gerakakis, Dieter Pfoser: **Utilizing Wireless Positioning as a Tracking Data Source**. In Proc. SSTD conf., pp. 171-188, 2009.

2 Objectives Change

In the original TALOS proposal it was stated that a Java 2 Mobile Edition (J2ME) application would be implemented that would use approximate positioning techniques for standard GSM networks and WiFi hotspots. Our contribution would lie in the development of a positioning technique, so that users will not need specialized hardware (i.e. GPS) in order to know their location. At that time, our contribution was state of the art, since only two companies on a global scale offered a similar solution (Skyhook Wireless and Ekahau).

However, when our work on Context Aggregation started, WiFi positioning technologies were already widely available. Platforms like Apple's iPhone and Google's Android incorporated the services of Skyhook Wireless, and applications like Google Gears and Firefox 3.5 Location Services were offering Wireless Positioning Services through an API. The semi-ubiquitous and game-changing nature of Apple and Google product offerings demanded a rethink of our original proposal towards the benefit of the project and the participating SMEs.

Therefore, in order to create a competitive edge for the SMEs, we shifted our attention towards the creation of a Wireless Positioning Service for WiFi hotspots that can be applied in both outdoors and indoors scenarios and that will allow the client's autonomous operation. As far as indoors position estimation is concerned, it allows the user to find her precise location within a museum or a shopping mall. At the time of writing, none of the aforementioned platforms comes with such capabilities. On the other hand, autonomous offline operation, provides another significant benefit for the SMEs, since existing positioning services require an Internet connection, which is typically not available for tourists in a foreign country due to forbidding (roaming) data tariffs.

Furthermore, both Apple's iPhone and Google's Android were paradigm-shifting devices and showed a new direction for the market of mobile devices in terms of capabilities (large touchscreens, integrated app stores, UI design). Therefore we moved away from the initially chosen Java 2 Mobile Edition platform and chose Apple's iPhone as our implementation platform judging it the most stable and promising at that time. Our decision was confirmed by the direction the mobile market is heading to (Nokia, SonyEricsson, Motorola), the success of Apple’s mobile platform (iPhone 3G/3GS), and the (still) unpolished nature of Google’s Android.

3 Evaluation of Approximate Positioning Techniques

Wireless Positioning Systems (WPS) provide a position estimate based on the radio signals received at a given location (*measurement*), and a known *radio map* of the environment. For example, in the case for 802.11 (WiFi) wireless networks, the measurement consists of a set of the visible access point ids (BSSID), and their corresponding *received signal strength* (RSS[1]). The measurement is then compared to the radio map through a *distance* metric, and a position estimate is calculated.

Different wireless positioning algorithms exist, which imply different forms and means to create the radio maps, as well as distance metrics to provide an estimate. In all cases, the radio maps for a given region are produced by training data, typically collected through *wardriving*. Wardriving is the process of massively collecting geocoded RSS measurements when driving through a certain geographic area. For a given measurement period (e.g., 5sec), we perform a scan of the available radio signals in the environment and obtain the position of this scan through GPS.

The vast majority of the WPS algorithms that can be found in the relative literature can be classified in one of the two following classes of WPS algorithms, centroid and fingerprinting. For both classes, numerous approaches and variations exist, depending on the wireless network (e.g., [8, 7, 9, 4, 10]) and environment (e.g., indoors/outdoors [6, 1, 2, 5]). We have either adopted these variations as is, or properly adapted and extended them to suit our case.

3.1 Centroid

Centroid is the simplest and the fastest method for wireless positioning. In centroid, the radio map consists of a set of the available Beacons and their positions, i.e. for WiFi, <*BSSID, X, Y*>. Consequently, centroid depends on having the true locations of the Beacon positions. Since this information is practically not available, nor feasible to produce, we must create the radio map from the training data, essentially *estimating* the position of the Beacons. Therefore, for each Beacon in the training data, we find all the positions it was visible, and estimate the Beacon's position as the arithmetic mean of these coordinates. Having established the radio map, a position estimate is provided in a similar manner. Given a measurement from the environment where certain Beacons are visible, we calculate the arithmetic mean of their coordinates, as provided by the radio map.

In order to improve accuracy when creating the radio map and/or calculating an estimate, we adopted *weighted* centroid from [3] and proposed two new heuristics: *k-max* and *thresholds*. Specifically:

Weighted. The simple arithmetic mean is substituted by a weighted arithmetic mean, where the weight is based on the RSS.

K-max. We apply the arithmetic mean on only the k Beacons with the lowest RSS (low RSS values correspond to strong received signal).

Thresholds. We define three thresholds $t_1 \leq t_2 \leq t_3$ which split the RSS space in four regions. If there are Beacons which fall in the first threshold ($RSS \leq t_1$), then we use only them in the arithmetic mean and ignore the rest. If there no Beacons in the first threshold, we use the ones in the second ($t_1 \leq RSS \leq t_2$), and so forth. In case there are Beacons only in the last threshold ($t_3 \leq RSS$), then

[1] Note that we always refer to the absolute value of RSS.

the algorithm does not provide an estimate since we consider the measurement to provide highly inaccurate readings.

Consequently, for centroid, there are a total of 16 different combinations of techniques to create the radio map and to provide an estimate: 4 to create the radio map, and 4 to provide an estimate. A specific centroid technique is be denoted as *centroid <radio map, estimation>*, where *radio map* and *estimation* can be one of the following: arithmetic mean (*am*), weighted (*w*), k-max (*k=n*), and thresholds (t_1-t_2-t_3). For example, centroid <k=2, 60-70-80>, means that the radio map was built with the *k-max* technique with k=2, and the estimation is provided with the *thresholds* technique with t_1=60, t_2=70, and t_3=80.

3.2 Fingerprinting

Fingerprinting assumes that the Beacons and associated RSS observed at a particular location are stable over time. Consequently, a measurement at a given location, i.e., the list of visible Beacons and RSS, can be considered as the *unique* fingerprint of that location. Thus, in fingerprinting, the training data themselves comprise the radio map.

To estimate the position, the algorithm calculates the Euclidean distance in the signal strength space between the current fingerprint and all available fingerprints in the radio map that contain the same Beacons. It then selects the *k-nearest* fingerprints in terms of distance, and returns as an estimate the arithmetic mean of their coordinates. This comparison is possible only if the current fingerprint and the fingerprints in the radio map contain exactly the same Beacons. Otherwise, calculating their distance in the Euclidean space is not possible.

However, in realistic conditions the current fingerprint may not contain exactly the same Beacons with the ones in the radio map. For example in the case of WiFi, some of the APs may have been turned off or removed, new APs may have been deployed, or the network interface may not provide APs with an RSS below a given threshold.

To account for this situation, we calculate the distance between the current fingerprint and the ones in the radio map based on a *subset* of common Beacons. In particular, we extended the algorithm in [6] so that the subset is defined by two parameters:

l: We compare the current fingerprint with fingerprints that contain *at most l less* Beacons. For example, suppose the WiFi scan <(AP_1, RSS_1), (AP_2, RSS_2), (AP_3, RSS_3)>. For l=1, a fingerprint <x_a, y_a, (AP_1, RSS_1), (AP_2, RSS_2)> would be included in the position estimation, in contrast with <x_b, y_b, (AP_2, RSS_2)> which would be ignored since there are two missing APs.

m: We compare the current fingerprint with fingerprints that contain *at most m more* APs. For example, suppose the WiFi scan <AP_1, RSS_1>. For m=1, the fingerprint <x_a, y_a, (AP_1, RSS_1), (AP_2, RSS_2), (AP_3, RSS_3)> would be excluded from the estimation due to the two extra APs.

Consequently, fingerprinting is modified as follows. The algorithm calculates the Euclidean distance in the signal strength space between the current fingerprint and all fingerprints in the radio map that contain at most l less and m more APs. It then selects the *k-nearest* fingerprints in terms of distance in the signal space, and returns as an estimate the arithmetic mean of their coordinates. As a result, there are many instances of the fingerprinting algorithm based on different parameters of *k*, l, and m. We use the notation *fingerprinting <k, l, m>* to denote a specific instance of the fingerprinting algorithm.

3.3 Evaluation

3.3.1 Experimental Setup

We evaluated the aforementioned Wireless Positioning techniques in terms of both accuracy and coverage through extensive experimentation.

3.3.2 Data Collection

For outdoors evaluation we collected data in the Zografou neighborhood in Athens, Greece. The area was selected (i) due to its geographical characteristics (mix of flat areas and hills), (ii) varying levels of WiFi AP density (0-15 APs/m^2), (iii) typical urban structure with a mix of shops and residential areas, and (iv) fluctuating traffic.

Data was collected through wardriving over a period of two months in an area covering approximately 100,000m^2. For data collection typical road speeds and driving habits were maintained. Driving speeds varied from 0kph (stationary for more than 5mins) to 70kph. Concerning the chosen wardriving approach, instead of multiple passes from each road segment (which may reveal more APs, produce more samples for an AP, etc.), we performed at most one pass. This implies that the collected data set may be less complete than it could be, but resembles a realistic *large scale* mapping effort to create the radio map of any given region.

For indoors evaluation we chose as experimentation locations two work areas. They were selected due to the different characteristics of their WiFi networks topology, few access points per floor for the first one and many access points in various locations for the second.

In contrast to an outdoors scenario, where the creation of an extensive and detailed radiomap is prohibitive due to the large geographical areas to be covered, the small size of an indoors location allows the increase of the sampling rate during data collection up to a several fingerprints per square meter, thus providing a complete radiomap of the environment.

3.3.3 Outdoors Evaluation

The following interesting observations can be made with respect to the data collected in our outdoors experimentation. First, the total number of unique APs discovered was 2,184, and on average we observed 5 APs for each sampled location. Considering the covered geographic area, this yields 2.1 APs per 100m^2. Second, in most cases when WiFi was not available, then GPS was not available as well (e.g., under a bridge, near a large building). Third, almost all APs were available 24/7. Overall, these facts confirm the increased penetration of WiFi networks in urban environments and constitute a foundation for the proliferation of WiFi-based WPS as a ubiquitous and dependable alternative to GPS.

We used our training data to create the radio maps and the testing data to calculate the position estimates based on these maps. We experimented with all permutations of means described in 3.1 and 3.2. For each point in the testing data, the position estimate provided by each WPS algorithm for specific parameter settings is compared to the respective GPS measurement taken (ground truth).

Table 1 shows the results concerning *accuracy* using a ranking based on the average error of the WPS estimates. In addition, for each result its respective coverage (i.e., the percent of times the technique can provide an answer) is stated. For each class of WPS algorithms (centroid, fingerprinting) the best three accuracy achieving parameter settings are presented. What can be observed is that given the right parameters, fingerprinting achieves the best positioning accuracy (25.24m). However, the results overall only differ slightly. What is of interest is the respective coverage that can be achieved with each method. For example, the best performing fingerprinting method has a coverage of 56%, i.e., the technique cannot provide a position estimate 44% of the time. This behavior is caused by the WPS algorithms themselves and by our wardriving approach to collect training data. For example, centroid<k=1, 60-80-90> provides an estimate based only on APs with RSS below 60. The estimate will be more accurate because the required RSS threshold is low, but since this is also highly selective, there are many instances where RSS below 60 is not available.

Table 1. WPS accuracy compared to GPS

	Average Error (m)	Coverage (%)
Fingerprinting <6-1-5>	**25.24**	**56**
Fingerprinting <6-1-4>	26.40	54
Fingerprinting <6-1-6>	26.57	56
Centroid <k=1, 60-70-80>	26.61	74
Centroid <k=1, 65-80-80>	26.65	82
Centroid <k=1, 75-85-90>	26.82	64

Table 2 ranks WPS techniques based on their *coverage* values. As expected, the techniques producing the best coverage underperform in terms of average error. To design an actual wireless positioning system one needs to consider this trade-off between accuracy and coverage, i.e., is providing a more accurate estimate better than always providing a crude estimate?

Table 2. WPS coverage

	Average Error (m)	Coverage (%)
Centroid <k=1, weighted>	35.52	94
Centroid <k=1, 70-80-85>	47.11	93
Centroid <k=1, 65-75-80>	47.15	92
Fingerprinting <6-2-6>	36.45	82
Fingerprinting <2-4-6>	51.53	81
Fingerprinting <6-6-1>	48.93	78

One conclusion to the above question is to provide a *hybrid* WPS technique for centroid and fingerprinting. In particular, we obtain an estimate from the best performing technique in terms of accuracy, but should the said technique not be available (coverage), we obtain an estimate from the technique with the best coverage. These hybrid WPS techniques have high coverage (> 96%) with an acceptable increase in average error (cf. Table 3).

Table 3. Average error and coverage of the hybrid WPS techniques

	Average Error (m)	Coverage (%)
Hybrid Centroid	32.77	99
Hybrid Fingerprinting	28.40	96

3.3.4 Indoors Evaluation

In an indoors environment GPS derived location is not available since GPS signal is blocked from buildings. Therefore in order to estimate the user's position a different reference system is required. In our evaluation we used the floor plan of each location. During data collection, each time a sample was taken, the user marked his current location on the floor plan. The estimated position calculated during the evaluation was then compared to the user provided location, using that as the "ground truth".

The detailed radiomap available in indoors environments assures that we have 100% coverage in all cases. Moreover, relevant literature indicates that given the dense radiomap containing fingerprints for practically every position, the technique that best exploits this information providing more accurate results is Fingerprinting [1, 3]. Centroid approximates the position using the locations of visible APs, which are rather sparse when compared to the density of the fingerprint sampling. Therefore, even in its weighted permutation, it fails to fully exploit the extra information provided by the detailed radiomap of the environment. Fingerprinting on the other hand, based on a similarity search between the current measurement and all the available fingerprints in the radiomap can provide very accurate results. The more the fingerprints, the more chances are to find several very close to the current measurement, thus indicating their location as the correct one.

Our evaluation also showed that the increased accuracy of the fingerprinting algorithm heavily depends on the setup of the indoor environment. Table 4 shows for each location the average error in meters of the Fingerprinting algorithm.

Table 4. Average indoors error

	Average Error (m)
Location 1	~3m
Location 2	~6m

At the first location were the evaluation took place there are only 1-2 Access Points per floor. Most of the fingerprints in the produced radiomap contained mainly these APs with relatively weak signal. Taking into consideration that, as it is indicated in the literature, RSS's decrease is not linear to the distance from the Access Point and that more distant measurements provide less credible conclusions about their distant from the AP (see Figure 1), we can see that most locations in a distance from the APs will provide similar fingerprints.

In contrast, at the second location many Access Points are available in different locations, covering all the areas with various signal strengths, depending on their distance from each location. This setup produced a radiomap with a wide variety of fingerprints and with each fingerprint having several APs with strong signal, thus being very characteristic of its position.

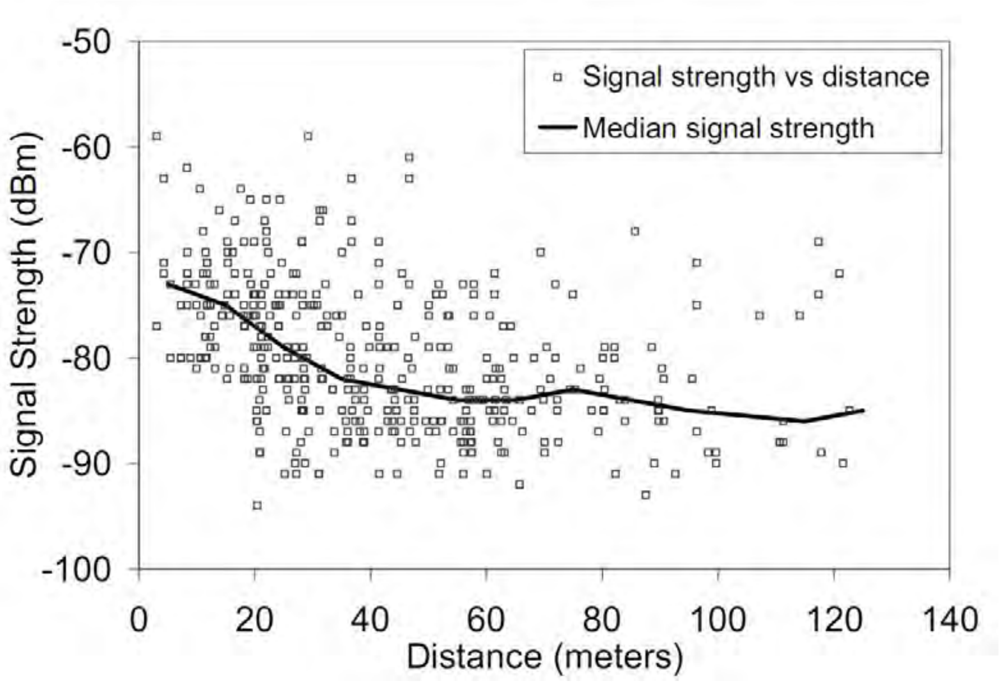


Figure 1. Measured RSS a afunction of the distance from AP

3.4 Conclusions

The evaluation of the approximate Wireless Positioning Techniques has shown that in outdoors scenarios the Fingerprinting algorithm provided better accuracy with centroid being a very close second. Taking into account though the significantly increased computational complexity of the fingerprint algorithm and the usually limited computational resources of mobile devices, centroid is the best choice for outdoors.

As far as indoors environments are concerned, our evaluation showed fingerprinting is the algorithm that can achieve the desired accuracy.

4 Architecture and Implementation

4.1 Architecture

The applications implemented in the context of Work Package 1 can by divided in two packages:

Approximate Positioning Tools. A set of applications was implemented in order to assist the creation of the necessary radio map, either by wardriving process in outdoors scenarios or for detailed mapping in indoor scenarios.

Context Aggregator. The iPhone library that provides contextual attributes to applications.

4.1.1 Approximate Positioning Support Tools

The Approximate Positioning Support Tools consist of the following components:

iwscand. Iwscand is responsible for gathering the information about the detectable Access Points and their RSS and accomplishes that by using the Linux Wireless Extensions [12] library. It also retrieves the current GPS coordinates by contacting the GPS daemon [13].

IwscanClient. A Library responsible for communicating with iwcand, parsing the supplied xml information and supplying it as in memory objects.

WPSurvey. This is the main application, a GUI that allows the user to create the radiomaps for the Wireless Positioning process. It uses *iwscanclient* to connect to *iwscand* and retrieve the necessary information.

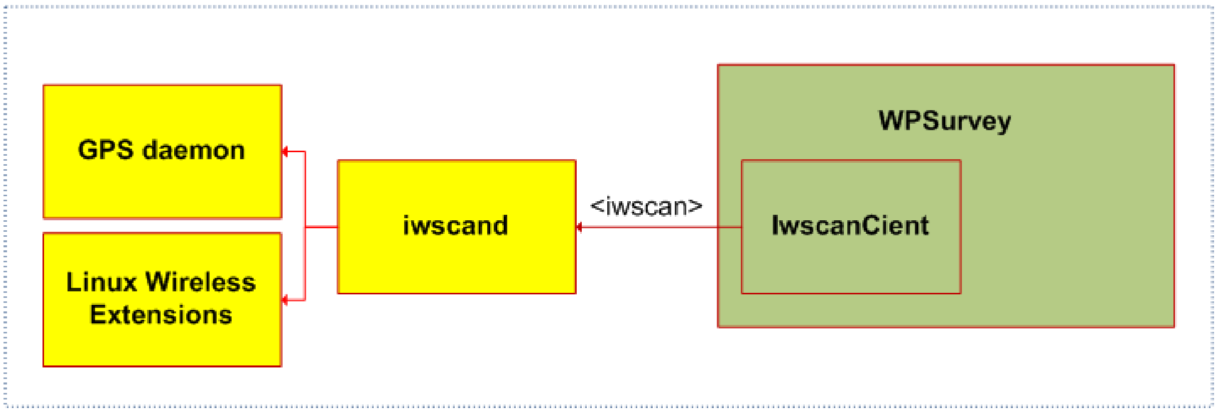


Figure 2. Approximate Positioning Support Tools Architecture

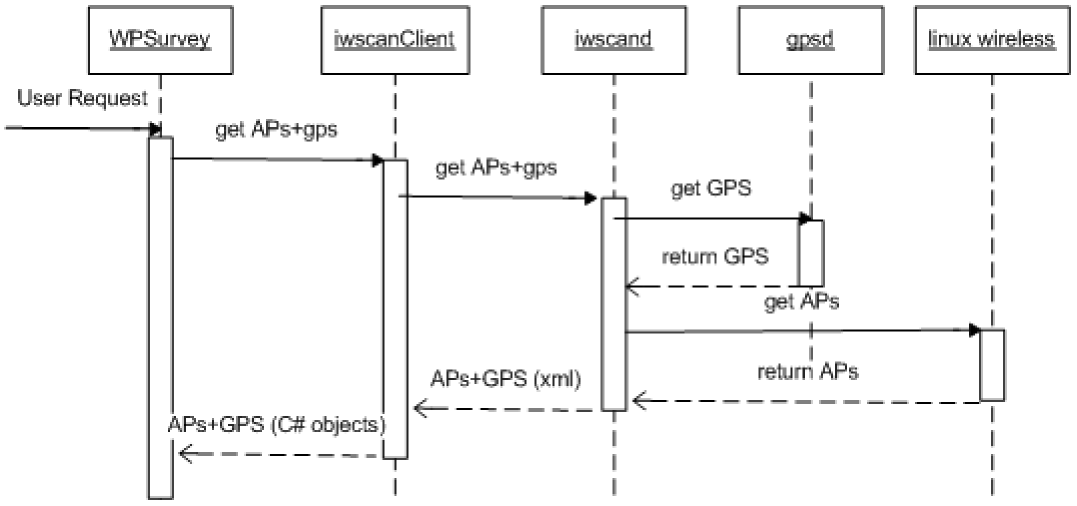


Figure 3. Approximate Positioning Support Tools Message exchange

4.1.2 Context Aggregator

Context's Aggregator consists of the following components:

ContextAggregator: This is the main class that provides the contextual attributes to requesting applications. In order to retrieve the necessary attributes it uses either existing components of the iPhone framework or one of the following components.

iWPS. The library that implements the Approximate Wireless Positioning operations in the mobile client.

CTXTSettings. The component responsible for storing and retrieving user and application settings.

WeatherProvider. This component retrieves weather forecast information from the web.

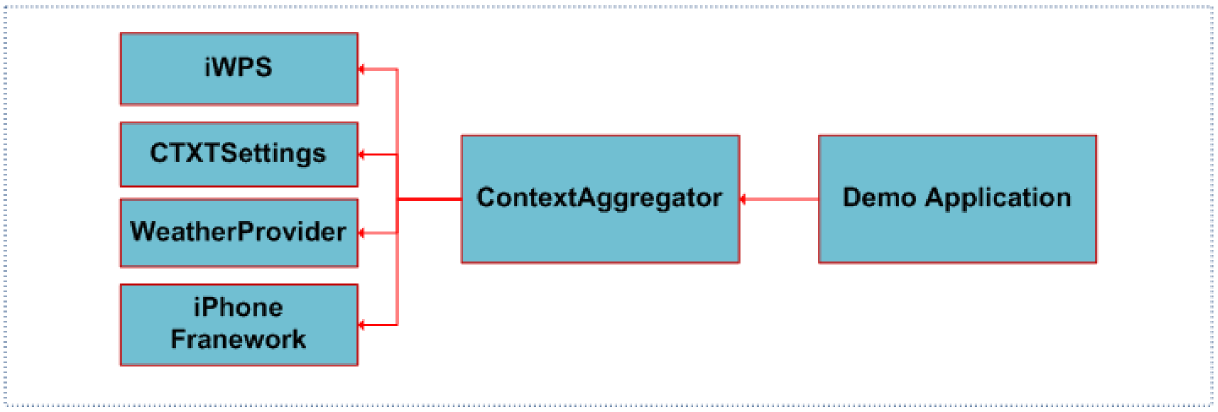


Figure 4. Context Aggregator Architecture

4.2 Implementation Details

All the Approximate Positioning Support Tools were implemented for the Linux Platform due to the unrestricted and standardized access to the information regarding the wireless connection. Though Iwscand which was implemented in C, IwscanClient and WPSurvey components were implemented in C# using the Mono Framework[14] and the Gtk# library[15] which enable the portability of these components to the Windows platform.

All Context Aggregator’s components were implemented in Objective-C for the iPhone Framework 3.1.2. It should be noted that due to the fact that the iPhone platform does not allow the use of external libraries or background processes the Context Aggregator’s source code must be incorporated in to the source of the TALOS iPhone application.

5 Components

In this section the Context's Aggregator components are presented in more detail. Complete description of the implementation for its component can be found in the documentation of the source code.

5.1 Approximate Positioning Support Tools

5.1.1 Iwscand

Iwscand runs as daemon that listens to a user specified port and upon requests returns the gather WiFi and GPS information in xml format. Below you can see an example of the returned xml document:

```
<iwscan>
    <ap essid="AP1" bssid=" 00:15:56:D0:1A:CB"
          qual="50" qualMax="100"
          noise="-10" noiseMax="0"
          signal="-70" singalMax="0"/>
    <ap essid="AP2" bssid=" 00:14:C1:1E:50:86"
          qual="70" qualMax="100"
          noise="-10" noiseMax="0"
          signal="-61" singalMax="0"/>
    <gps latitude="37.9989" longitude="23.76657” altitude="194.7”
</iwscan>
```

daemon is listening at using a TCP socket and send the message "*pos*" followed by a new line character.

When called, iwscand uses the first command line argument as port it should listen to.

5.1.2 IwscanClient

IwscanClient is used as a C# library (dll) so that other C# applications can easily retrieve the information provided by Iwscand. In order to avoid unnecessary multiple connections from the application to iwscand it initiates a shared static instance which requests information for GPS and WiFi scans and return it as an Iwscan C# in memory object.

Applications can access the aforementioned functionality by calling the following static method:

```
Iwscan IwscanClient.scan()
```

IwscanClient read its configuration using C#'s CofigurationManager's App.config xml file. More specifically, applications using IwscanClient must have the following settings defined in the `<appSettings>` section of their App.config:

`iwscand-address`: The IP address iwscand is listening to.

`iwscand-port`: The port iwscand is listening to.

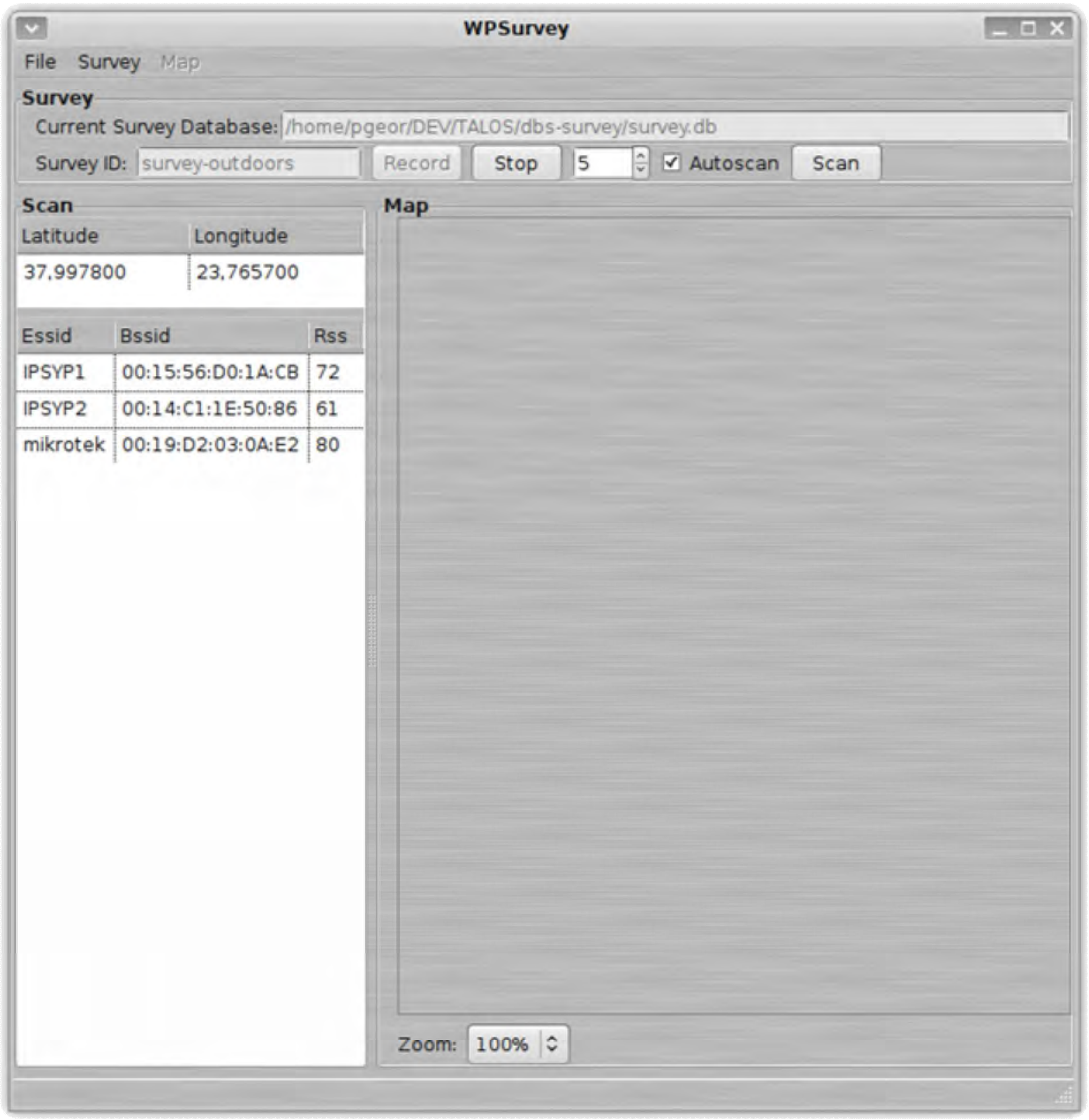
`iwscand-timeout`: Timeout waiting for iwscand responce (in milliseconds).

An example App.config file follows:

```
<appSettings>
    <add key="iwscand-address"  value="127.0.0.1"/>
    <add key="iwscand-port"     value="5000"/>
    <add key="iwscand-timeout"  value="2000"/>
</appSettings>
```

5.1.3 WPSurvey

WPSurvey is the GUI users can use in order to scan the environment for WiFi networks, combine the measurements with their current location and create the radiomap that will be used by the positioning component of the mobile application.



Current Survey Database: /home/pgeor/DEV/TALOS/dbs-survey/survey.db

Survey ID: survey-outdoors

5 ☑ Autoscan

Latitude	Longitude
37.997800	23.765700

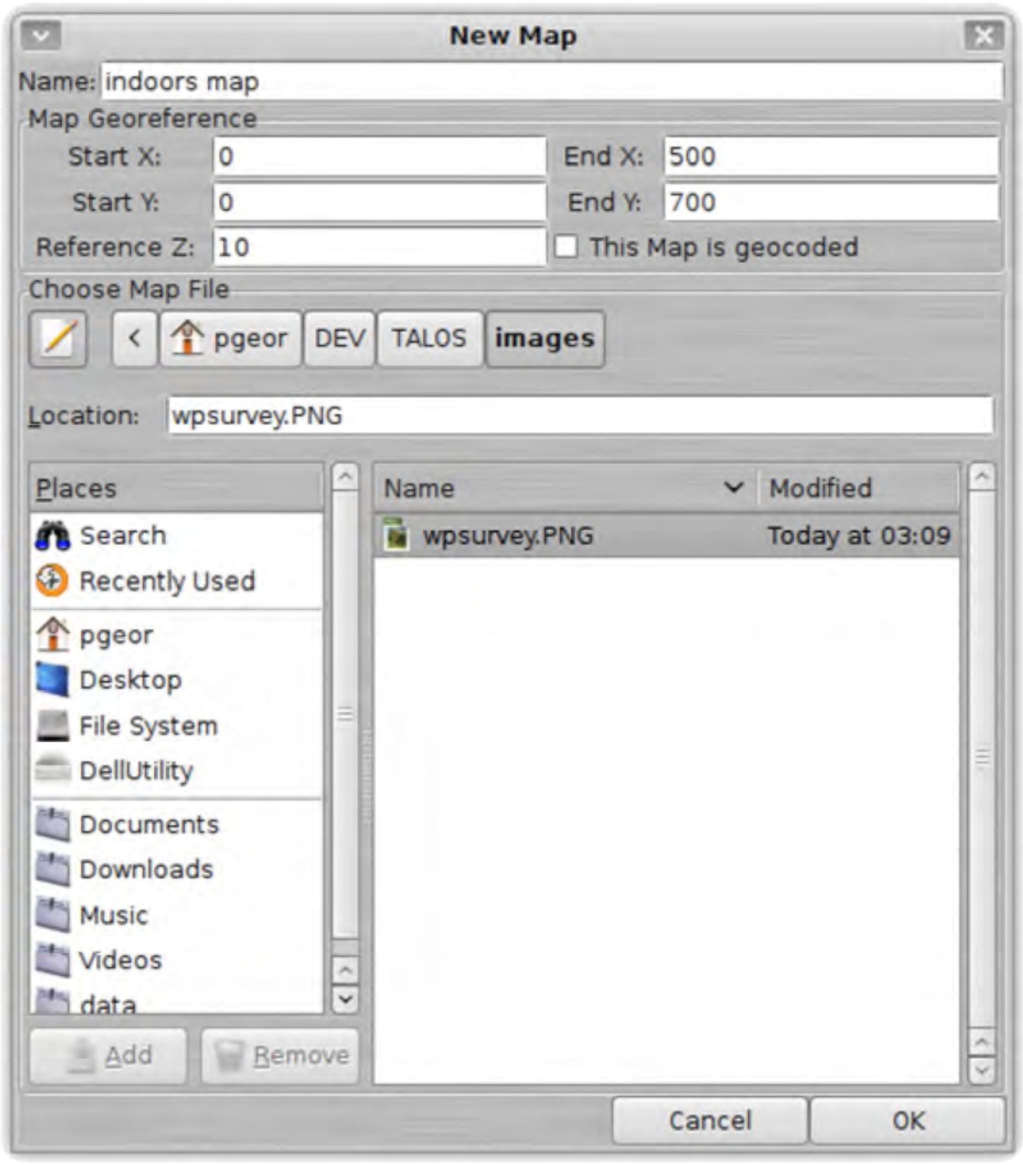
Essid	Bssid	Rss
IPSYP1	00:15:56:D0:1A:CB	72
IPSYP2	00:14:C1:1E:50:86	61
mikrotek	00:19:D2:03:0A:E2	80

Zoom: 100%

Figure 5. WPSurvey - Outdoors mapping

Data collections are logically organized "Surveys". The collected data from various surveys is stored in a user defined Sqlite [16] database. The data can then be processed and exported to another Sqlite database compatible with the schema used by the Approximate Positioning component running in the mobile device (iPhone).

There are two types of surveys available, outdoors and indoors. Outdoors surveys require the existence of GPS coordinates in each acquired scan from iwscand in order for them to be recorded. They also support automatic scanning in regular user defined time interval. Finally when exporting data to a database to be used in the iPhone it uses the centroid algorithm in order to estimate the positions of the available Access Points. All three variations of the centroid algorithm have been implemented and which ne i used is configurable from the WPSurvey settings file. Figure 5 shows a screenshot of the main window of WPSurvey for an outdoors survey.


New Map

Name: indoors map

Map Georeference

Start X: 0 End X: 500

Start Y: 0 End Y: 700

Reference Z: 10 ☐ This Map is geocoded

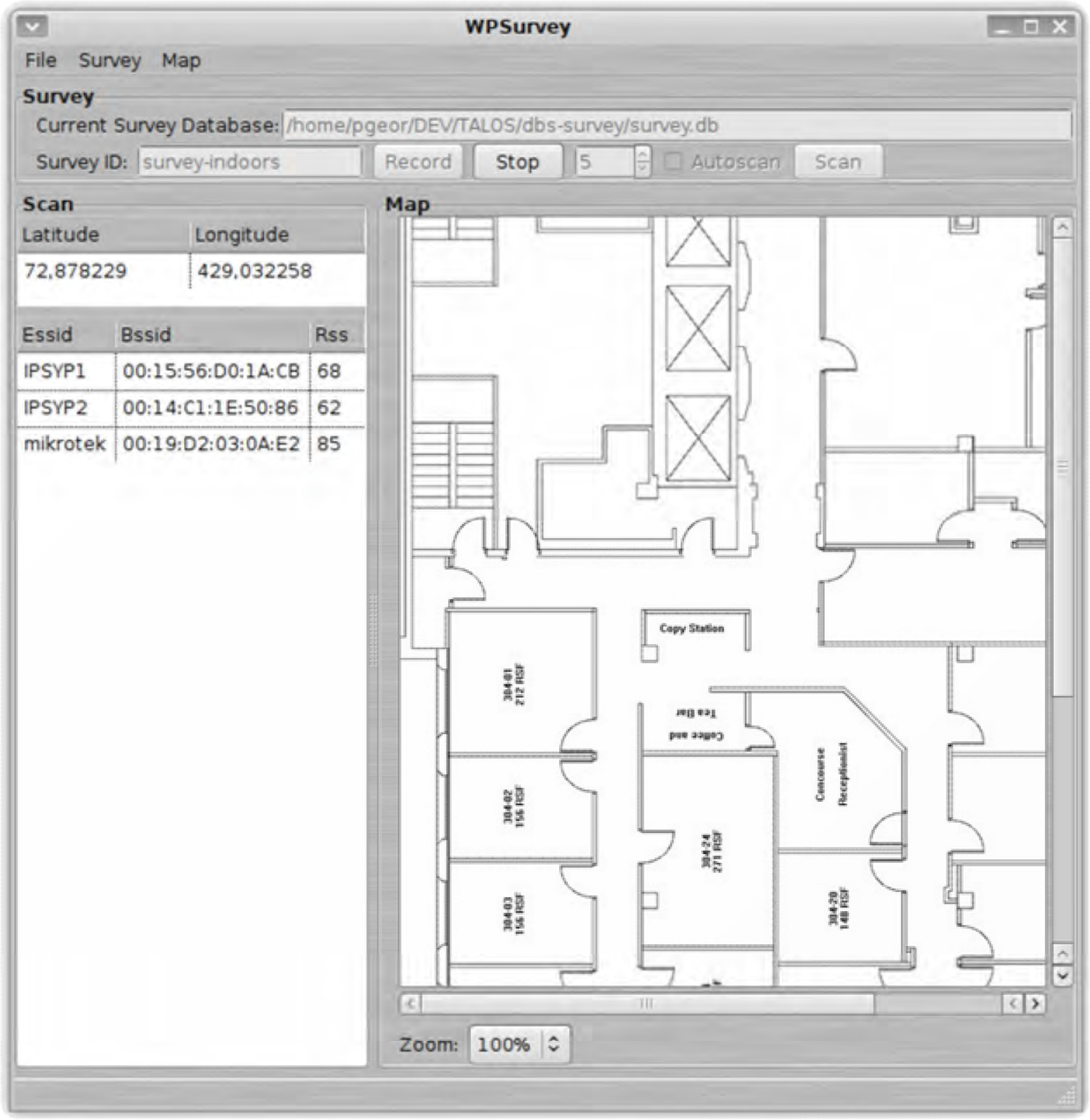
Choose Map File

Location: wpsurvey.PNG

Name	Modified
wpsurvey.PNG	Today at 03:09

Figure 6. WPSurvey – New map

Indoors surveys on the other hand do not require the use of GPS coordinates since GPS signal is not available indoors. Instead the specification of a map (usually a floor plan) of the environment is required. For ease of use and for greater compatibility the map is provided as a Portable Network Graphics (PNG) picture. The user must specify reference coordinates for the provided map (see Figure 6). These coordinates need not be real GPS coordinates, they can be an arbitrary reference system, but it required that they follow the standard geographical orientation, that means the point (startX, startY) is the left lower point of the map. The provided map is saved in the database and a lower resolution version of it is exported to the iPhone format database along with the data. When exporting indoors survey's data no processing is applied, the fingerprints are simply copied into the destination database since the Fingerprinting technique uses the raw radiomap.In indoors surveys, in order for the user to perform a scan of the environment he must click on a point on the provided map. The acquired scan is saved with its coordinates set as the coordinates of map point the user clicked as derived from the maps reference system. Figure 7 is a screenshot of the main window of WPSurvey for an indoors survey.



Survey

Current Survey Database: /home/pgeor/DEV/TALOS/dbs-survey/survey.db

Survey ID: survey-indoors

5

Scan

Latitude	Longitude
72,878229	429,032258

Essid	Bssid	Rss
IPSYP1	00:15:56:D0:1A:CB	68
IPSYP2	00:14:C1:1E:50:86	62
mikrotek	00:19:D2:03:0A:E2	85

Zoom: 100%

Figure 7. WPSurvey – Indoors mapping

WPSurvey reads the necessary configuration data using using C#'s CofigurationManager's App.config xml file. Apart from the required settings for the IwscanClient library that were explained in 5.1.2. the following settings must be defined:

`CentroidMapCreationMethod`: The Centroid permutation used when exporting the radiomap for the Approximate Positioning library. The possible values are:

- o `kmax`
- o `thresholds`
- o `weighted`

`CentroidMapKmax`: The value of the *k* parameter of the *k-max* permutation of the centroid technique.

`CentroidMapThreshold1`, `CentroidMapThreshold2`, `CentroidMapThreshold3`: The thresholds used by the *threshold* permutation of the centroid technique.

`iPhoneMapImageSizeX`, `iPhoneMapImageSizeY`: The size to which the map's image is scaled down to when exported to the iPhone database.

`logfile`: The files were the application logs its operations.

`DEBUG-LEVEL`: Defines the verbosity of the applications log file. 0 stands for errors only, values below 10 increase the verbosity, 10 means all operations.

`DefaultOpenPath`: Defines the default path that is prompted to the user in Open/Save operations.

An example App.config file that includes the parameters for IwscanClient follows:

```
<appSettings>
    <add key="iwscand-address"              value="127.0.0.1"/>
    <add key="iwscand-port"                 value="5000"/>
    <add key="iwscand-timeout"              value="2000"/>
    <add key="CentroidMapCreationMethod"    value="kmax"/>
    <add key="CentroidMapKmax"              value="3"/>
    <add key="CentroidMapThreshold1"        value="60"/>
    <add key="CentroidMapThreshold2"        value="70"/>
    <add key="CentroidMapThreshold3"        value="80"/>
    <add key="iPhoneMapImageSizeX"          value="280"/>
    <add key="iPhoneMapImageSizeY"          value="320"/>
    <add key="logfile"                      value="WPSurvey.log"/>
    <add key="DEBUG-LEVEL"                       value="0"/>
    <add key="DefaultOpenPath"              value="/home"/>
</appSettings>
```

5.2 Context Aggregator

5.2.1 ContextAggregator

The ContextAggregator class offers an single point access to all the available contextual information provided from the other components. The information is returned either as return values of ContextAggregator's methods or by calling the appropriate "delegates" implemented by the requesting class. The following context is offered by the Context Aggregator:

Current Location from the iWPS component. Relevant methods:

- startUpdatingLocation()
- stopUpdatingLocation()

Current weather and weather forecast from the Weather Component. Relevant methods:

- updateWeather()
- getForecastForDay(date)

User or Application Settings. Relevant methods:

- getAppSettingForKey(key)
- getUserSettingForKey(key)
- setUserSettingForKey(key)

Current Date and Time. Relevant methods:

- getDate()
- getTime()
- getDateTime()

Settings from the mobile device, like the system model, version or name. Relevant methods:

- getDeviceName()
- getSystemType()
- getSystemModel()

Some operations performed by the Context Aggregator take some time to finish and return their results. Such operations are the position approximation that needs to perform a scan for reachable WiFi APs or any network related operations like getting the latest weather forecast. In order to avoid UI blocking until these operations finish, they are implemented so that they run in their own thread and upon completion they return the acquired result by calling a specific method of the delegate object as that has been defined by the caller. The methods that the delegate object's class must implement in order to receive the proper message are defined in an interface (protocol in Objective-C). So, in order for a class to be eligible as delegate for a specific operation it must implement the relative interface – protocol.

In our implementation two delegate protocols are used:

`CLLocationManagerDelegate`: This is the standard delegate protocol used in iPhone's framework for communicating with the built-in positioning services. We chose to use the same protocol since a "familiar" protocol makes the integration and use of the Context Aggregator much easier.

`WeatherDelegate`: This is the protocol that needs to be implemented in order to receive information about the Weather forecast.

5.2.2 iWPS

This component is the largest component of the Context Aggregator. It uses the techniques for Approximate Positioning using WiFi networks discussed in section 3 in order to estimate the user's current position. The required radiomap is stored in a Sqlite database as it exported from the WPSurvey tool. The database can be either distributed with the application or downloaded from the web and used offline.

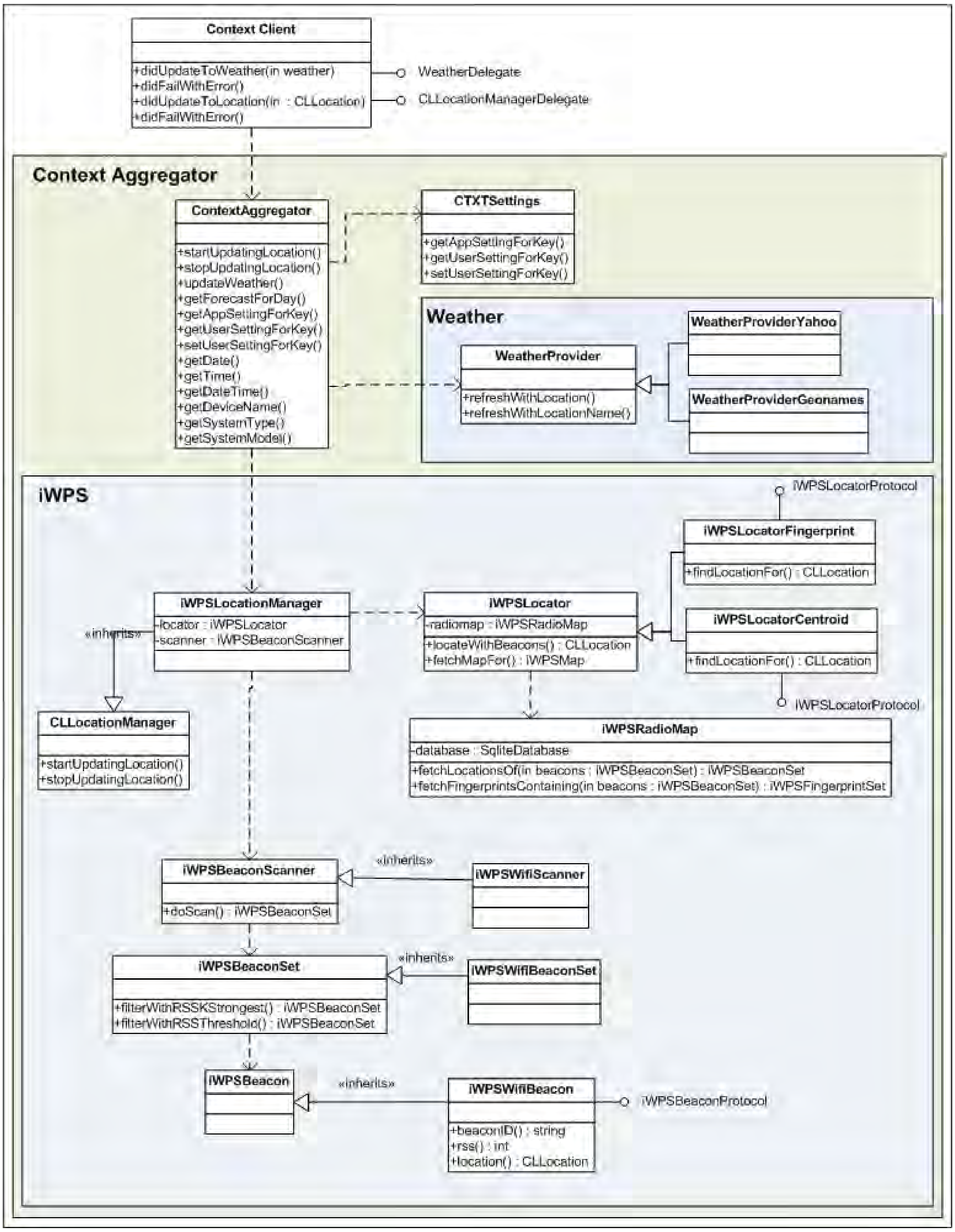


Figure 8. Context Aggregator – Class diagram

In order to facilitate the use of the Context Aggregator component and its integration into applications the implementation tries to offer familiar interfaces. Therefore the standard classes defined in the Core Location API of iPhone's Framework were used. The component's main class `iWPSLocationManager` extends the iPhone's native `CLLocationManager`. Also, as we explained in the previous section, the delegate protocol used is `CLLocationManagerDelegate`, again being a native iPhone protocol. This behaviour enables any application using iPhone's Positioning libraries to switch to iWPS simply by instantiating our own Location Manager instead of the native one.

iWPS library uses WiFi networks in order to find the current location. However the capability to acquire the location from iPhone's built-in GPS hardware is offered if the desired accuracy for the iWPS library is set to the value `kCLLocationAccuracyNearestTenMeters`. Moreover, as it is illustrated in Figure 8 which shows the class diagram for the Context Aggregator Components, a flexible architecture has been adopted that allows the extension of the component for any other type of wireless network.

5.2.3 Weather

Weather information is retrieved from the free web service that is provided by GeoNames [17] and returns the current weather for a given set of coordinates. Using the current location provided from the positioning library the current weather is returned without the need for any user input. In order to demonstrate the flexibility of the Context Aggregator's implementation and its independency from the weather provider, we have also implemented another weather provisioning service, this is from Yahoo![18].

5.2.4 Settings

This component is responsible for the retrieval of any Application or User setting. Application settings are distributed with the applications; they are stored in an xml file (iPhone's plist) and have read-only access. User settings on other hand are saved in the mobile device and can be changed at any time.

5.2.5 Demo Application

To demonstrate the operation of the Context Aggregator we have implemented a demo iPhone application that retrieves context and displays it on the device's screen. Figures 9 and 10 show screenshots of the application when displaying an outdoors and an indoors position.

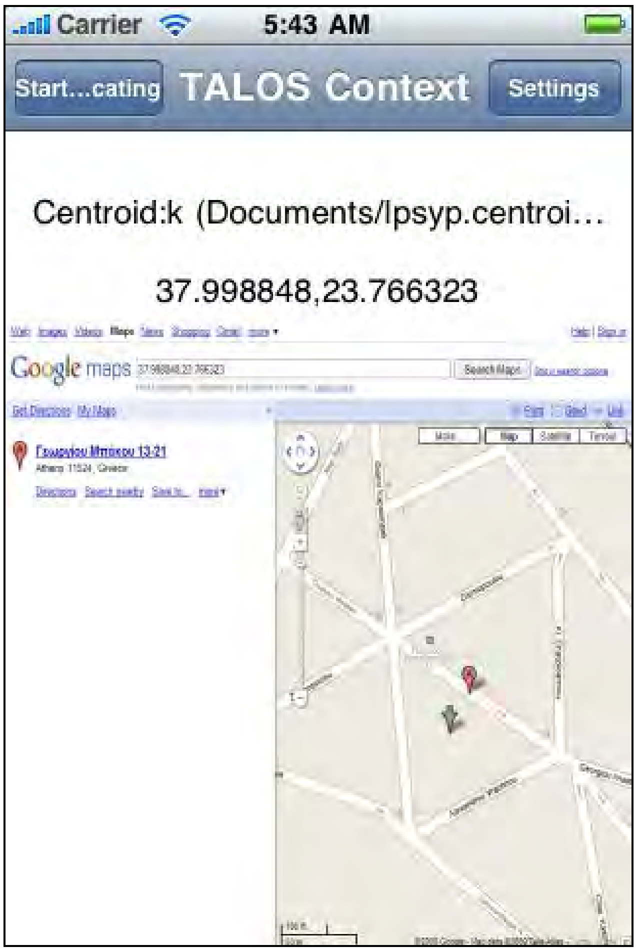


Figure 9. Demo Application - outdoors

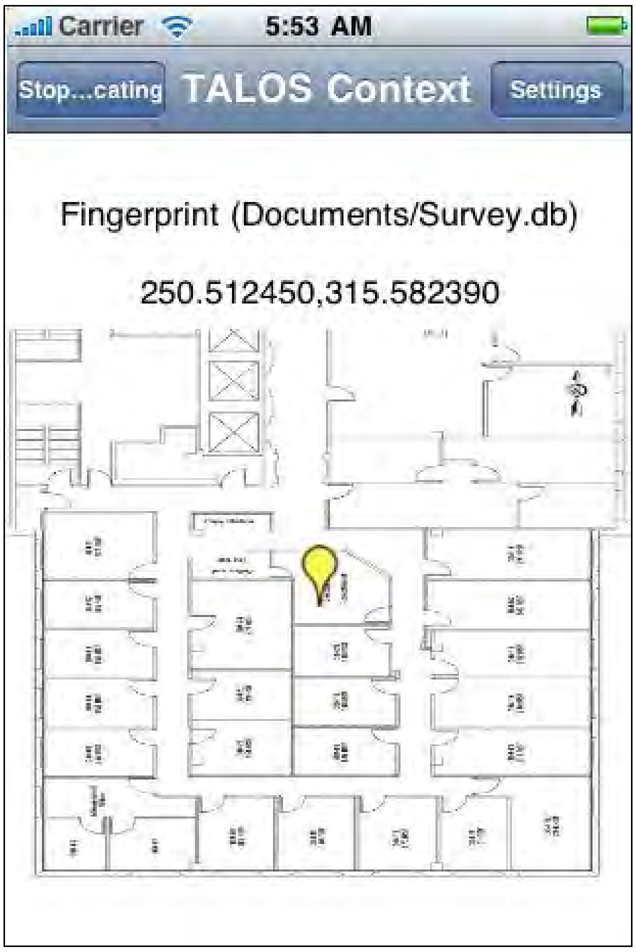


Figure 10. Demo Application - indoors

Conclusions

A significant effort was spent in evaluating the current positioning techniques. An experimental approach was adopted allowing extensive evaluation in realistic everyday scenarios. The best performing algorithms in terms of accuracy, coverage and performance were chosen and implemented in the Approximate Positioning Library.

The original Work Package objectives were revised and altered in order to preserve the project’s competitive edge. Focus shifted to WiFi networks, indoors positioning and innovative mobile devices like the iPhone.

A Context Aggregator was developed as a software component that captures the user’s contextual attributes. Several subcomponents have also been implemented in order to provide the required information. A “soft-GPS” component was created that enables the estimation of the current position without the need for a specialized hardware (i.e. GPS) or the requirement for an Internet Connection, in both indoors and outdoors environments. A suite of tools was created to support the approximate positioning operations.

References

1. Bahl P., Padmanabhan V.N.: RADAR: An In-Building RF-Based User Location and Tracking System. In : 9th IEEE Conference on Computer Communications, pp. 775-784, IEEE Press (2000)
2. Bahl, P., Padmanabhan, V.N., Balachandran, A.: Enhancements to the Radar User Location and Tracking System. Technical Report, Microsoft Research MSR-TR-00-12 (2000)
3. Cheng, Y., Chawathe, Y., LaMarca, A., Krumm, J.: Accuracy Characterization for Metropolitan-scale Wi-Fi Localization. In: 3rd International Conference on Mobile Systems, Applications, and Services, pp. 233–245, ACM(2005)
4. Chen, M.Y., Sohn, T., Chmelev, D., Hightower, D.H.J., Hughes, J., LaMarca, A., Potter, F., Smith, I., Varshavsky, A.: Practical metropolitan-scale positioning for GSM phones. In: 8th International Conference on Ubiquitous Computing. LNCS, vol. 4206, pp. 225-242, Springer (2006)
5. Hightower, J., Consolvo, S., LaMarca, A., Smith, I., Hughes, J.: Learning and recognizing the places we go. In: 7th International Conference on Ubiquitous Computing. LNCS, vol. 3660, pp. 105-122, Springer (2005)
6. Krishnan, P., Krishnakumar, A.S., Ju, W., Mallows, C., Ganu, S.: A system for LEASE: Location estimation assisted by stationary emitters for indoor RF wireless network. In: 23rd IEEE Conference on Computer Communications, pp. 1001-1011, (2004)
7. LaMarca, A., Chawathe, Y., Consolvo, S., Hightower, J., Smith, I., Scott, J., Sohn, T., Howard, J., Hughes, J., Potter, F., Tabert. J., Powledge. P., Borriello. G., Schilit. B.: Place Lab: Device Positioning Using Radio Beacons in the wild. In: 3rd International Conference on Pervasive Computing. LNCS, vol. 3468, pp. 116-133, Springer (2005)
8. Laitinen, H., Lahteenmaki. J., Nordstrom. T.: Database correlation method for GSM location. In Proceedings of the 53rd IEEE Vehicular Technology Conference, pp. 2504-2508, IEEE Press (2001)
9. Otsason, V., Varshavsky, A., LaMarca, A., Lara, E.D.: Accurate GSM Indoor Localization, in Proceedings of Ubicomp. LNCS, vol. 3660, pp. 141-158, Springer (2005)
10. Sohn, T., Varshavsky, A., LaMarca, A., Chen, M.Y., Choudhury, T., Smith, I., Consolvo, S., Hightower, J., Griswold, W.G., Lara, E.D.: Mobility Detection Using Everyday GSM Traces. In: 8th International Conference on Ubiquitous Computing. LNCS, vol. 4206, pp. 212-224, Springer (2006)
11. Varshavsky, A., Chen, M., Lara, E.D., Froehlich, J., Haehnel, D., Hightower, J., LaMarca, A., Potter, F., Sohn, T., Tang, K., Smith, I.: Are GSM phones THE solution for localization?. In: 7th IEEE Workshop on Mobile Computing Systems and Applications, pp. 20-28, IEEE Press (2006)
12. http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Tools.html
13. http://gpsd.berlios.de/
14. http://mono-project.com
15. http://www.mono-project.com/GtkSharp
16. http://www.sqlite.org/
17. http://www.geonames.org/
18. http://weather.yahoo.com/

Appendix

Spiros Athanasiou, Panos Georgantas, George Gerakakis, Dieter Pfoser: **Utilizing Wireless Positioning as a Tracking Data Source**. In Proc. SSTD conf., pp. 171-188, 2009.

Utilizing Wireless Positioning as a Tracking Data Source

Spiros Athanasiou[1], Panos Georgantas[2], George Gerakakis[2], and Dieter Pfoser[1,3]

[1] Institute for the Management of Information Systems
"Athena" Research Center
G. Mpakou 17, 11524Athens, Greece
{spathan, pfoser}@imis.athena-innovation.gr

[2] School of Electrical and Computer Engineering
National Technical University of Athens
Greece, 15780
{pgeor, ggera}@dblab.ece.ntua.gr

[2] RA Computer Technology Institute
Dvaki 10
Greece, 11524
pfoser@cti.gr

Abstract. Tracking data has become a valuable resource for establishing speed profiles for road networks, i.e., travel-time maps. While methods to derive travel time maps from GPS tracking data sources, such as floating car data (FCD), are available, the critical aspect in this process is to obtain amounts of data that fully cover all geographic areas of interest. In this work, we introduce Wireless Positioning Systems (WPS) based on 802.11 networks (WiFi), as an additional technology to extend the number of available tracking data sources. Featuring increased ubiquity but lower accuracy than GPS, this technology has the potential to produce travel time maps comparable to GPS data sources. Specifically, we adapt and apply readily available algorithms for (a) WPS (centroid and fingerprinting) to derive position estimates, and (b) map matching to derive travel times. Further, we introduce map matching as a means to improve WPS accuracy. We present an extensive experimental evaluation on real data comparing our approach to GPS-based techniques. We demonstrate that the exploitation of WPS tracking data sources is feasible with existing tools and techniques.

Keywords: wireless positioning, map matching, tracking, FCD

1 Introduction

Incorporating travel times into road network information, i.e., travel time maps, is an important prerequisite for a large number of spatiotemporal tasks. Examples include shortest path computation, traffic avoidance, emergency response, etc.

Solutions typically rely on collected floating car data (FCD) that sample the overall traffic conditions [16, 5] in a given region. FCD capture temporal variations in achievable vehicle speeds throughout the road network. For example, speeds during the rush-hour are considerably lower than during night traffic. Then, in a post-processing step termed map-matching [4, 19], tracking data is accurately related to the road network and travel times are extracted. It is critical that *large amounts* of FCD are available for long periods of time and geography, so that the extracted speed profiles are accurate. Currently, all methods use GPS for tracking the position of vehicles.

1.1 The case for GPS vs. WPS

While GPS is the most popular positioning technique, it has several drawbacks. First, it requires the use of specific hardware limiting the number of vehicles or users that can collect and provide tracking data. Second, there are occasions where GPS is inadequate (e.g., limited coverage, interference of high frequency electronic equipment). This is especially true for "urban canyons", i.e., areas in urban environments where line-of-sight with the GPS satellites is obscured, leading to inaccurate readings or no coverage at all. As demonstrated by LaMarca et al. [11], the average availability of GPS in an urban environment is only 4.5% during a user's daily schedule. In contrast, wireless networks, such as WiFi and GSM, are available on average 94.5% and 99.6% respectively. Third, the addition of extra integrated or autonomous GPS modules lead to increased power consumption, and thus limit the user's mobility or application of GPS.

These drawbacks of GPS have led to the rise of Wireless Positioning Systems (WPS), where the user location is estimated with the help of other, readily available wireless networks. As a technology, WPS delivers less accurate results (e.g., ~40m for WiFi/outdoors), but provides greater coverage characteristics (e.g., above 90% of a user's time). Further, WPS can be integrated in practically any computing device that incorporates a wireless network interface, and with a negligible burden on the interface's power consumption. So while WPS is less accurate than GPS, for typical everyday applications it can efficiently augment or even replace GPS.

Lately, WPS capable devices and applications are becoming a common place for end users, with examples like the iPhone, Android, Google Gears, Mozilla Firefox 3.1, etc. In addition, the integration of WPS in GPS and WiFi chipsets (e.g., SiRF, Broadcom, Texas Instruments) will result in a state where practically all mobile devices will have WPS capabilities. This argument is a fact, rather than a prediction, with great implications on spatiotemporal data management in general. In combination with the emerging usage of geolocation Web APIs (e.g., W3C Geolocation) we anticipate that in the near future there will be an abundance of readily available WPS positioning data.

Consequently, the technical advance of WPS is leading to new challenges and potential gains for numerous applications, where the scale and amount of positioning data will require corresponding advances in algorithmic solutions. Further, repurposing this sort of data by accommodating their particularities (e.g., varying levels of accuracy, ubiquitous coverage, etc.) in order to extract hidden knowledge, will be another area of great interest.

Our work is therefore extremely relevant in this newly established context, and applied to the specific issue of creating travel time maps. Currently, the creation of travel time maps from actual travel data is based solely on FCD. While this guarantees the use of position readings of high accuracy, it also limits the availability of such data for extended periods of time and geography. However, by successfully exploiting WPS, we would have access to data (a) whose size is several orders of magnitude greater, (b) temporally span bigger periods, and (c) extend to larger geographic areas. One could argue that WPS is only feasible in urban areas. While this observation is true, it actually strengthens our argument; urban areas are *exactly* where travel time maps are valuable resources for routing solutions.

1.2 Contributions

In this work, we advocate the use of WPS to complement and/or replace GPS tracking data sources to produce travel time maps. This increases the potential number of data providers and ultimately the quality of the resulting travel times. To the best of our knowledge, this is the first attempt of repurposing WPS tracking data to produce travel time maps. Our contributions are:

- We adapt and extend the two most important classes of WPS algorithms (centroid and fingerprinting) for our setting (WiFi network, outdoors operation).
- We experimentally evaluate the optimal parameters of the various classes of WPS algorithms and identify an optimal solution in terms of accuracy and coverage under realistic settings.
- We adapt an online map-matching algorithm to WPS tracking data as a post-processing step to improve WPS accuracy.
- We adapt a global map-matching algorithm to extract travel time maps from historic WPS tracking data and compare the results to GPS derived travel time maps.
- We demonstrate that for high sampling frequencies, WPS derived travel times are comparable to GPS in absolute terms. Further, even for low sampling frequencies, the results in terms of speed profiles (categories) are useful as well.

The remainder of this paper is structured as follows. Section 2 introduces techniques for wireless positioning. Section 3 briefly introduces the map-matching algorithm used for deriving travel times from tracking data. Section 4 gives an experimental evaluation of WPS techniques and travel times derived from WPS data. Finally, Section 5 presents our conclusions and directions for future research.

2 Wireless Positioning

Wireless Positioning Systems (WPS) provide a position estimate based on the radio signals received at a given location (*measurement*), and a known *radio map* of the environment. In the case for 802.11 (WiFi) wireless networks, the measurement consists of a set of the visible access point ids (BSSID), and their corresponding *received signal strength* (RSS[1]). The measurement is then compared to the radio map through a *distance* metric, and a position estimate is calculated.

Different wireless positioning algorithms exist, which imply different forms and means to create the radio maps, as well as distance metrics to provide an estimate. In all cases, the radio maps for a given region are produced by training data, typically collected through *wardriving*. Wardriving is the process of massively collecting geocoded RSS measurements when driving through a certain geographic area. For a given measurement period (e.g., 5sec), we perform a scan of the available WiFi networks in the environment (BSSID, RSS) and obtain the position of this scan through GPS.

In this section, we present the outline of two classes of WPS algorithms we adapted and implemented for our experiments, i.e., centroid and fingerprinting. For both classes, numerous approaches and variations exist, depending on the wireless network (e.g., [13, 12, 17, 7, 18]) and environment (e.g., indoors/outdoors [10, 2, 3, 8]). We have either adopted these variations as is, or properly adapted and extended them to suit our case.

2.1 Centroid

Centroid is the simplest and the fastest method for wireless positioning. In centroid, the radio map consists of a set of the available APs and their positions, i.e., *<BSSID, X, Y>*. Consequently, centroid depends on having the true locations of the AP positions. Since this information is practically not available, nor feasible to produce, we must create the radio map from the training data, essentially *estimating* the position of the APs. Therefore, for each AP in the training data, we find all the positions it was visible, and estimate the AP's position as the arithmetic mean of these coordinates. Having established the radio map, a position estimate is provided in a similar manner. Given a measurement from the environment where certain APs are visible, we calculate the arithmetic mean of their coordinates, as provided by the radio map.

In order to improve accuracy when creating the radio map and/or calculating an estimate, we adopted *weighted* centroid from [6] and proposed two new heuristics: *k-max* and *thresholds*. Specifically:

Weighted. The simple arithmetic mean is substituted by a weighted arithmetic mean, where the weight is based on the RSS.

K-max. We apply the arithmetic mean on only the k APs with the lowest RSS (low RSS values correspond to strong received signal).

Thresholds. We define three thresholds $t_1 \leq t_2 \leq t_3$ which split the RSS space in four regions. If there are APs which fall in the first threshold ($RSS \leq t_1$), then we use only them in the arithmetic mean and ignore the rest. If there no APs in the first threshold, we use the ones in the second ($t_1 \leq RSS \leq t_2$), and so forth. In case there are APs only in the last threshold ($t_3 \leq RSS$), then the algorithm does not provide an estimate since we consider the measurement to provide highly inaccurate readings.

[1] Note that we always refer to the absolute value of RSS.

Consequently, for centroid, there are a total of 16 different combinations of techniques to create the radio map and to provide an estimate: 4 to create the radio map, and 4 to provide an estimate. A specific centroid technique will be denoted as *centroid <radio map, estimation>*, where *radio map* and *estimation* can be one of the following: arithmetic mean (*am*), weighted (*w*), k-max (*k=n*), and thresholds (t_1-t_2-t_3). For example, centroid <k=2, 60-70-80>, means that the radio map was built with the *k-max* technique with k=2, and the estimation is provided with the *thresholds* technique with t_1=60, t_2=70, and t_3=80.

2.2 Fingerprinting

Fingerprinting assumes that the APs and associated RSS observed at a particular location are stable over time. Consequently, a measurement at a given location, i.e., the list of visible APs and RSS, can be considered as the *unique* fingerprint of that location. Thus, in fingerprinting, the training data themselves comprise the radio map.

To estimate the position, the algorithm calculates the Euclidean distance in the signal strength space between the current fingerprint and all available fingerprints in the radio map that contain the same APs. It then selects the *k-nearest* fingerprints in terms of distance, and returns as an estimate the arithmetic mean of their coordinates. This comparison is possible only if the current fingerprint and the fingerprints in the radio map contain exactly the same APs. Otherwise, calculating their distance in the Euclidean space is not possible.

However, in realistic conditions the current fingerprint may not contain exactly the same APs with the ones in the radio map. For example, some of the APs may have been turned off or removed, new APs may have been deployed, or the network interface may not provide APs with an RSS below a given threshold (typical behavior of Windows 802.11 hardware drivers).

To account for this situation, we calculate the distance between the current fingerprint and the ones in the radio map based on a *subset* of common APs. In particular, we extended the algorithm in [6] so that the subset is defined by two parameters:

l: We compare the current fingerprint with fingerprints that contain *at most l less* APs. For example, suppose the WiFi scan <(AP_1, RSS_1), (AP_2, RSS_2), (AP_3, RSS_3)>. For *l*=1, a fingerprint <x_a, y_a, (AP_1, RSS_1), (AP_2, RSS_2)> would be included in the position estimation, in contrast with <x_b, y_b, (AP_2, RSS_2)> which would be ignored since there are two missing APs.

m: We compare the current fingerprint with fingerprints that contain *at most m more* APs. For example, suppose the WiFi scan <AP_1, RSS_1>. For *m*=1, the fingerprint <x_a, y_a, (AP_1, RSS_1), (AP_2, RSS_2), (AP_3, RSS_3)> would be excluded from the estimation due to the two extra APs.

Consequently, fingerprinting is modified as follows. The algorithm calculates the Euclidean distance in the signal strength space between the current fingerprint and all fingerprints in the radio map that contain at most *l* less and *m* more APs. It then selects the *k-nearest* fingerprints in terms of distance in the signal space, and returns as an estimate the arithmetic mean of their coordinates. As a result, there are many instances of the fingerprinting algorithm based on different parameters of *k*, *l*, and *m*.

During the rest of the paper, we will use the notation *fingerprinting <k, l, m>* to denote a specific instance of the fingerprinting algorithm.

3 Map Matching

Deriving travel times from tracking data implies the alignment of the tracking data with a respective trajectory in the road network, i.e., finding the actual roads the vehicle has traversed. Now, provided that the tracking data is precise, this task would be simple. However, tracking data is obtained by sampling a vehicle's movement, typically with GPS and in our case with WPS. Unfortunately, both *GPS and WPS are not precise* due to the *measurement error* caused by the limited positioning accuracy, and the *sampling error* caused by the sampling rate, i.e., not knowing where the moving object was in between position samples [14]. Therefore, a processing step is needed that matches tracking data to the road network. This technique is commonly referred to as *map matching*.

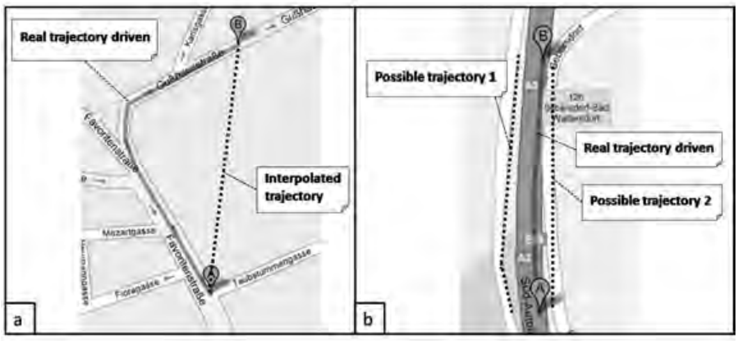


Fig. 1. Map-Matching example.

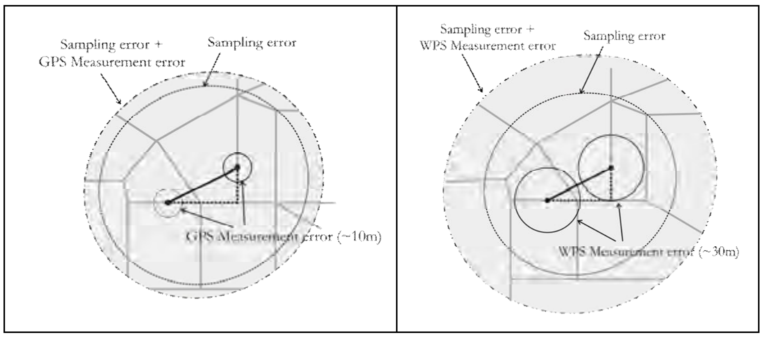

Sampling error + GPS Measurement error
Sampling error
GPS Measurement error (~10m)
Sampling error + WPS Measurement error
Sampling error
WPS Measurement error (~30m)

Fig. 2. Sampling error and measurement error.

To illustrate these errors and the map-matching problem in general, Fig.1 gives two examples of measured positions and the possible trajectory the vehicle could have taken. Fig. 1a shows the interpolated path in between position samples A and B and the actual path with respect to the road segment. Further, as evident in Fig.1b, the positioning error becomes significant when facing several parallel roads close by. Specifically, in the case of WPS (Fig.2), the measurement error might grow quite large. This significantly increases the challenge for proper map-matching, since with a large measurement error, one is presented many alternative paths in the road network to map the sampled movement to. Thus, we expect that at least minimizing the sampling error by using high sampling rates will prove to be important.

3.1 Theoretical Considerations

Most map-matching algorithms are tailored towards mapping *current* positions onto a vector representation of a road network. Onboard systems for vehicle navigation utilize, besides continuous positioning, dead reckoning to minimize the positioning error and to produce accurate positions that can be easily matched to a road map. However, for the purpose of processing tracking data collected over a period of time, the entire trajectory, given as a sequence of historic position samples, needs to be mapped.

The algorithm we utilize in this work is the *global map-matching algorithm* of [4, 19], which employs the Fréchet distance measure for curves [1]. A popular illustration of the Fréchet distance is the following. Suppose a person is walking his dog, the person is walking on the one curve and the dog on the other. Both are allowed to control their speed but they are not allowed to go backwards. The *Fréchet distance* of the curves is the minimal length of a leash that is necessary for both to walk the curves from beginning to end. Using this distance measure, our global map-matching algorithm tries to match the tracking data geometry to a respective path in the road network by comparing it to the shapes of all possible paths in the road network. Although conceptually quite an elaborate task, this can be accomplished in $O(mn\log mn)$ time, with m being total number of nodes and edges of the road network and n the size of the tracking data to be matched [4].

The global map-matching algorithm is therefore a *shape-matching algorithm* that matches one curve, the tracking data trajectory, to another curve, the road network path that most closely resembles the tracking trajectory. As such, the algorithm is predestined for matching historic data.

Consider now the *online map matching* case, in which tracking data is matched as it is collected, i.e., in real time. Here, we apply the same global map matching algorithm, but instead of exploiting the complete trajectory (which is not known), we take advantage of the available historic data, i.e., the tracking data available so far. Experimentation showed that typically a trajectory consisting of 10 position samples collected with a sampling rate of 30s can be matched with the same accuracy as longer trajectories, i.e., 10 position samples represent a reasonably large enough curve for the global map-matching algorithm to produce a good quality match when applied to the online case. Hence, to perform online map matching, we apply the global map matching algorithm on the trajectory formed by the current position estimate and the 9 last position estimates.

3.2 Deriving Travel Times

Having mapped the tracking data to the road network, travel times are derived by mapping the travel times contained in the tracking data to the respective portions of the road network. The map-matching algorithm performs essentially shape matching and tries to find a path in the road network that most closely resembles the trajectory, i.e., the tracking data (cf. dotted line in Fig. 4). In the process, it maps all position samples (circles in Fig. 4) to the road network and all nodes along the corresponding path to the tracking data trajectory. Since the original tracking data contained the timestamp they were received, this information is transferred to the map-matched tracking data along the road network. The former can be seen as an effort to rediscover where on the road network the position samples would have been originally recorded. As such, these mappings are the ideal means for assigning travel times to the respective road network edges. Overall, the approach we employ is to uniformly map the time recorded between two consecutive position samples (e.g., t_{i+1} - t_i) in Figure 4, to the respective portions of the road network.

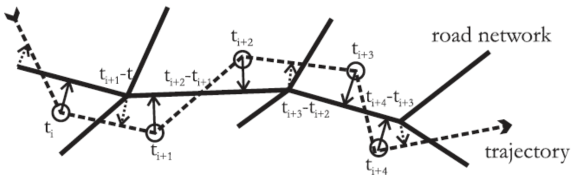


Fig. 3. Distance and travel time assignment.

4 Experimental Evaluation

The primary scope of our experimental evaluation is to establish the suitability of WPS data as a source to provide travel times. First, to provide a complete examination of the relevant technologies and potential uses, we provide an evaluation of WPS accuracy and coverage and also introduce map matching as a means to improve WPS accuracy.

4.1 Experimental setup

The experimentation was carried out in the Zografou neighborhood of Athens, Greece. The area was selected (i) due its to geographical characteristics (mix of flat areas and hills), (ii) varying levels of WiFi AP density (0-15 APs/m^2), (iii) typical urban structure with a mix of shops and residential areas, and (iv) fluctuating traffic.

4.1.1 Data Collection

Data was collected through wardriving over a period of two months in an area covering approximately 100,000m^2. For data collection typical road speeds and driving habits were maintained. Driving speeds varied from 0kph (stationary for more than 5mins) to 70kph. Fig. 5 shows a respective map of the Zografou area and the sampled locations on the road network where at least one WiFi AP was visible.

Our data set consists of records of the form <*tid, x, y, t, AP*>, where *tid* is the unique id of a trajectory, *x and y* are the GPS coordinates, *t* is the timestamp of the measurement, and *AP* is a list of the APs (BSSID) and their respective received signal strength (RSS). The sampling rate during data collection (i.e., every when a measurement is taken from the environment) was 5sec. In total, we collected roughly 200MBs of data, and we divided them (70%-30%) into two separate sets: (a) the *training* data, which were used to create the maps for the WPS techniques, and (b) the *testing* data, which were used to assess the WPS accuracy and to calculate travel times.

Concerning the chosen wardriving approach, instead of multiple passes from each road segment (which may reveal more APs, produce more samples for an AP, etc.), we performed at most one pass. This implies that the collected data set may be less complete than it could be, but resembles a realistic *large scale* mapping effort to create the radio map of any given region.

The equipment that was used comprised an Intel Core Duo laptop with a single 802.11a/b/g NIC and two Bluetooth GPS devices, all situated in the passenger compartment. We used Kismet [9] with a set of custom add-ons to extract geocoded WiFi measurements. All wardriving logs were later offloaded to a PostGIS database. Our WPS algorithms (centroid, fingerprinting) were developed in C/C++ and the map-matching algorithm was implemented in Java. Certain auxiliary processing/visualization tools were developed in PHP, Python, and Java. Accurate map data for the road network of Zografou were provided by Eratosthenis S.A. The experiments were executed by three Windows 2000 servers over a period of two weeks. Visualization of the results was performed with QGIS [15].

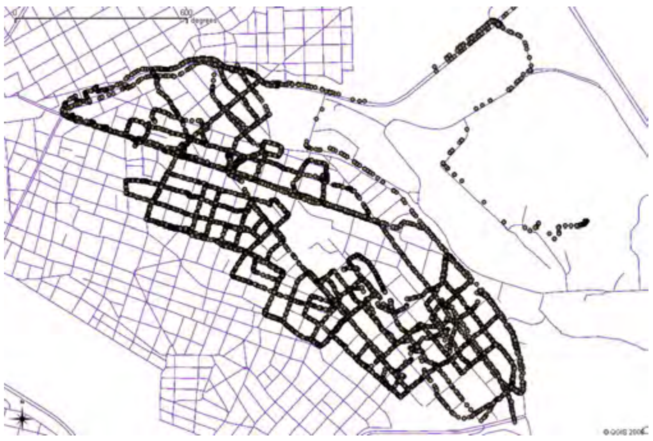


Fig. 4. Zografou map and WiFi AP locations.

4.1.2 WPS Feasibility

The following interesting observations can be made with respect to the data. First, the total number of unique APs discovered was 2,184, and on average we observed 5 APs for each sampled location. Considering the covered geographic area, this yields 2.1 APs per 100m^2. Second, in most cases when WiFi was not available, then GPS was not available as well (e.g., under a bridge, near a large building). Third, almost all APs were available 24/7. Overall, these facts confirm the increased penetration of WiFi networks in urban environments and constitute a foundation for the proliferation of WiFi-based WPS as a ubiquitous and dependable alternative to GPS.

4.2 WPS Positioning Accuracy

4.2.1 WPS accuracy and coverage

The following experimentation evaluates WPS techniques in terms of accuracy and coverage. In particular, we used our training data to create the radio maps and the testing data to calculate the position estimates based on these maps. We experimented with all permutations of means described in Section 2. For each point in the testing data, the position estimate provided by each WPS algorithm for specific parameter settings is compared to the respective GPS measurement taken (ground truth).

Table 1 shows the results concerning *accuracy* using a ranking based on the average error of the WPS estimates. In addition, for each result its respective coverage (i.e., the percent of times the technique can provide an answer) is stated. For each class of WPS algorithms (centroid, fingerprinting) the best three accuracy achieving parameter settings are presented. What can be observed is that given the right parameters, fingerprinting achieves the best positioning accuracy (25.24m). However, the results overall only differ slightly. What is of interest is the respective coverage that can be achieved with each method. For example, the best performing fingerprinting method has a coverage of 56%, i.e., the technique cannot provide a position estimate 44% of the time. This behavior is caused by the WPS algorithms themselves and by our wardriving approach to collect training data. For example, centroid<k=1, 60-80-90> provides an estimate based only on APs with RSS below 60. The estimate will be more accurate because the required RSS threshold is low, but since this is also highly selective, there are many instances where RSS below 60 is not available.

Table 1. WPS accuracy compared to GPS

	Average Error (m)	Coverage (%)
Centroid <k=1, 60-70-80>	26.61	74
Centroid <k=1, 65-80-80>	26.65	82
Centroid <k=1, 75-85-90>	26.82	64
Fingerprinting <6-1-5>	**25.24**	**56**
Fingerprinting <6-1-4>	26.40	54
Fingerprinting <6-1-6>	26.57	56

Table 2 ranks WPS techniques based on their *coverage* values. As expected, the techniques producing the best coverage underperform in terms of average error. To design an actual wireless positioning system one needs to consider this trade-off between accuracy and coverage, i.e., is providing a more accurate estimate better than always providing a crude estimate?

Table 2. WPS coverage

	Average Error (m)	Coverage (%)
Centroid <k=1, weighted>	35.52	94
Centroid <k=1, 70-80-85>	47.11	93
Centroid <k=1, 65-75-80>	47.15	92
Fingerprinting <6-2-6>	36.45	82
Fingerprinting <2-4-6>	51.53	81
Fingerprinting <6-6-1>	48.93	78

One conclusion to the above question is to provide a *hybrid* WPS technique for centroid and fingerprinting. In particular, we obtain an estimate from the best performing technique in terms of accuracy, but should the said technique not be available (coverage), we obtain an estimate from the technique with the best coverage. These hybrid WPS techniques have high coverage (> 96%) with an acceptable increase in average error (cf. Table 3).

Table 3. Average error and coverage of the hybrid WPS techniques

	Average Error (m)	Coverage (%)
Hybrid Centroid	32.77	99
Hybrid Fingerprinting	28.40	96

Unless stated otherwise, hybrid WPS techniques will be used through the rest of our experiments, denoted as *WPS-C* and *WPS-F* for hybrid centroid and hybrid fingerprinting respectively.

4.2.2 Map Matching to Improve WPS Accuracy

Map-matching is known as a technique to relate tracking data to a map dataset. One can also see it as a method for imposing geometric constraints (shapes of paths in the road network) to tracking data. As such, this technique might be a viable means to "correct" WPS data and improve its accuracy. In this experiment, we utilize two map-matching algorithms, a simple one (called *naive*) that maps position samples to the closest point on the road network and the online algorithm presented in Section 4.2, which exploits shape information. To compare the various approaches in terms of accuracy, we calculated the average error and standard deviation for the complete WPS dataset with respect to the GPS measurements.

The results are given in Table 4 and confirm the findings in the relevant literature, with fingerprinting providing more accurate results than centroid. However, note that in both cases the average error is roughly 30m. Further, while the naïve map-matching algorithm only marginally reduces the average error (~1m), the shape-based map-matching algorithm reduces the average error by 37% (WPS-C) and 25% (WPS-F). This happens, because in contrast to a naïve map-matching approach, the shape-based algorithm exploits past WPS estimates to produce a trajectory that best fits the road network. Hence, an extremely important side-effect of proper map-matching, stemming from its inherent robustness towards *inaccurate* data, is the improvement of the accuracy provided by WPS. Combining WPS with map-matching reduces the average error of WPS (~20m) very close to the average error of GPS in urban environments (5-15m). This observation clearly opens the room for more research and experimentation, since in the WPS literature GPS is always considered as the *ground truth* for calculating the average error. Obviously, this is something needed to be questioned given our findings. Our future work and current experimentation is focused on exploiting GNSS available in Greece of greater accuracy (<1m), such as Galileo CS [20] and HEPOS [21].

Table 4. WPS average error and standard deviation.

	Avg. Error (m)	Stdev. (m)	Avg. Error with naïve mm (m)	Stdev. with naïve mm (m)	Avg. Error with mm (m)	Stdev. with mm (m)
WPS-C	32.77	49.80	31.74	48.34	**20.47**	19.74
WPS-F	28.40	42.48	28.36	41.68	**21.15**	22.16

Moreover, we performed a set of experiments to assess the impact of the data collection speed, and AP density, towards WPS accuracy. In particular, to assess the impact of the data collection speed (i.e. frequency of collecting measurements from the environment), we removed records from the collected data to simulate frequencies ranging from 2Hz to 0,2Hz (Fig.5a). Further, we sampled our entire data set to randomly remove APs in order to simulate densities up to only 25% of the original one (Fig.5b). Our results illustrate that centroid is the most robust technique, maintaining an acceptable average error at all times.

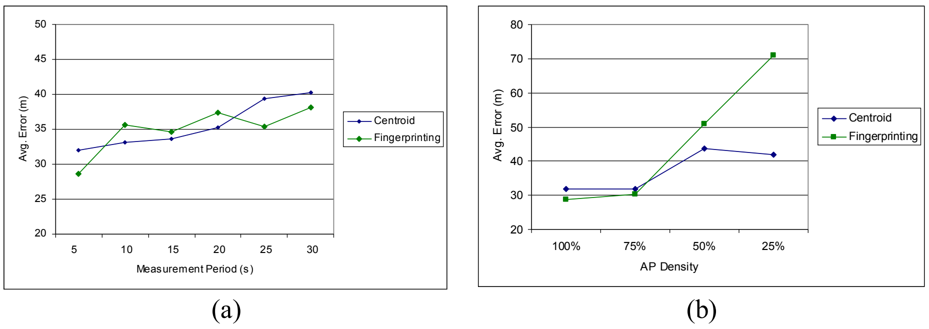


Fig. 5. Average error dependence from (a) measurement period and (b) AP density.

4.3 Extracting travel-time maps

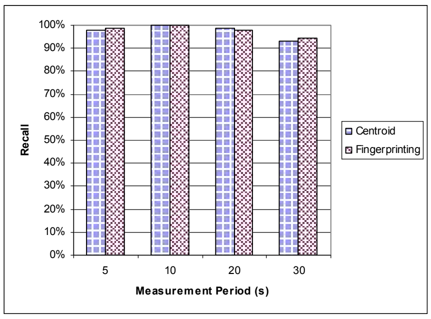
To establish the feasibility of using WPS data to derive travel times, we compared the travel times produced from GPS data to the ones produced from WPS data for the same trajectories. The format of the collected testing data was <*tid, x, y, t, AP*>, where *tid* the trajectory id, *x* and *y* the GPS coordinates, *t* the timestamp of the measurement, and *AP* the WiFi-related measurements, i.e., AP BSSIDs and RSS. For the testing data, WPS-C and WPS-F were used to produce WPS estimates, resulting in trajectory data of the form of <*tid, x, y, t, xc, yc, xf, yf*>, where *xc, yc, xf,* and *yf* are the coordinates produced by the centroid and fingerprinting algorithms respectively. For the three types of trajectory data, GPS, WPS-C, and WPS-F, global map-matching was applied, and using the approach detailed in Section 3, the respective *travel times* were derived for each case. Consequently, for each road segment in our network, we established three different travel time estimates, (i) GPS, (ii) WPS-C and (iii) WPS-F. Versions of the travel time dataset were produced for sampling rates of 5, 10, 20, and 30secs.

4.3.1 Qualitative evaluation

In order to compare the trajectories produced by GPS and WPS position data, we will define the measures of *recall* and *precision*. Let $G_i = \{g\}$, be the set of vertices produced by the map-matching algorithm on GPS data, for trajectory *i*. Also, let $W_i = \{w\}$ be the set of vertices produced by the map matching algorithm on WPS data for the same trajectory. The intersection $G_i + W_i$ contains the vertices the two sets have in common. Recall *R* and precision *P* can be defined as follows:

$$R_i = \frac{|G_i + W_i|}{|G_i|} \qquad P = \frac{|G_i + W_i|}{|W_i|} \qquad \textbf{(1)}$$

R indicates the fraction of road segments covered by GPS trajectories that is also covered by WPS. Ideally, *R* should be equal to 1, i.e., WPS returns all the road segments GPS does (but possibly more). Further, *P* indicates the fraction of road segments covered by WPS trajectories that is also covered by GPS. Again, we want *P* to be equal to 1, i.e., WPS does not produce road segments not produced by GPS.



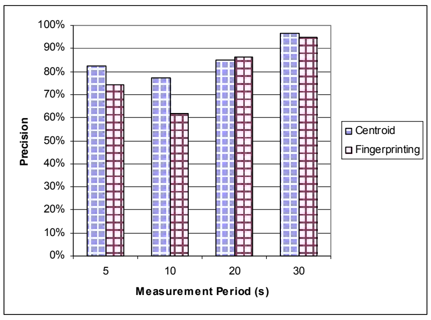
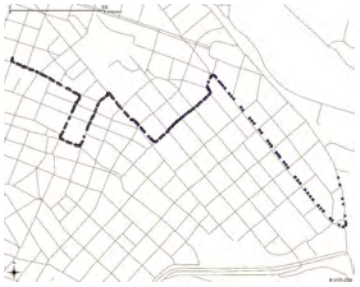
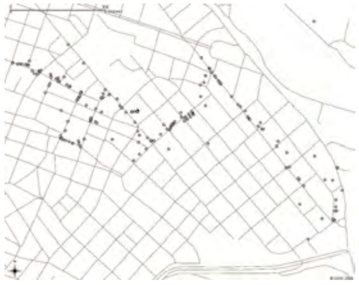


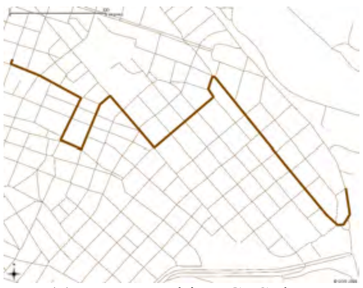
Fig. 6. Recall and precision for the WPS derived trajectories in our entire data set.

In Fig.6, the values of recall and precision for our entire data set using varying sampling rates are shown. Common to all cases, recall is high, close to 100%. Notice that recall is optimal for a sampling rate of 10s while precision is best for a sampling rate of 30s. This was expected, as for low sampling rates, the sampling error dominates the measurement error in the map matching process. Thus, *both WPS and GPS produce practically the same trajectories*.

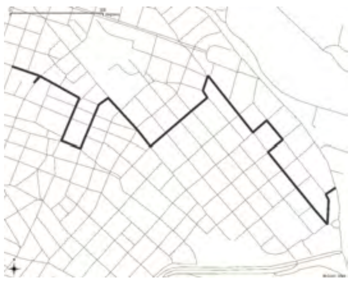
Fig.7 illustrates the above by giving a sample trajectory that accurately represents our findings for the entire data set. Fig.7(a) shows raw GPS tracking data while Fig.7(b) shows the WPS estimates derived by the WPS-C technique. Notice that although the 'noise' in WPS estimates is apparent (with several outliers as well), the trajectory can easily be distinguished. Fig.7(c),(d) show the produced trajectories after applying our map matching algorithm using a sampling rate of 30s. Fig. 7(e),(f) show details of the trajectory, highlighting specific map-matching cases.


(a) GPS data

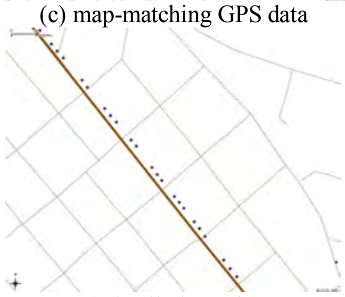

(b) Centroid WPS data



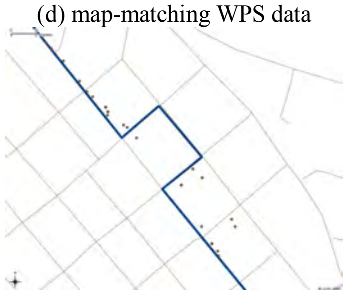
(c) map-matching GPS data



(d) map-matching WPS data



(e) detail view of (c)



(f) detail view of (d)

Fig. 7. Sample trajectory.

4.3.2 Quantitative Evaluation

Having established how trajectories produced by WPS fare in comparison to GPS, in the following, we compare the respective travel times derived from these approaches. Given the set of links for which WPS and GPS derived travel times are available, we calculated the average error of WPS compared to GPS derived travel times, as shown in Fig.8. What can be readily observed is that the optimal sampling period is 10s, with no real difference between the two WPS techniques. For a period of 30s, the errors are 80.3% (WPS-C) and 125.4% (WPS-F). This could be interpreted as a serious problem for map matching based on WPS data for lower sampling frequencies, since most travel time databases are calculated from fleet management logs with sampling periods of 20-30s.

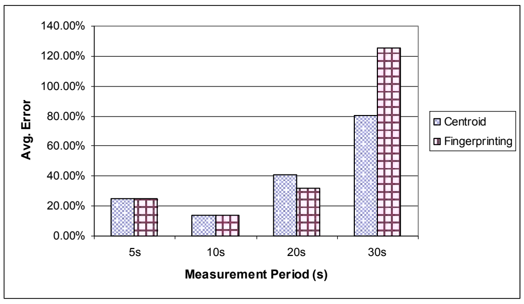


Fig. 8. Average error of WPS derived travel times compared to GPS derived travel times.

However, for the creation of *dynamic road network profiles*, travel times are used to classify road network links. For example, suppose that a road category is defined as including speeds ranging from 10-20kph. Here two road links with respective travel times of 10.5 and 19.5kph will be subsumed under the same category. This *quantization* is beneficial, because it results to lower storage requirements, faster route calculation, and routes of similar quality.

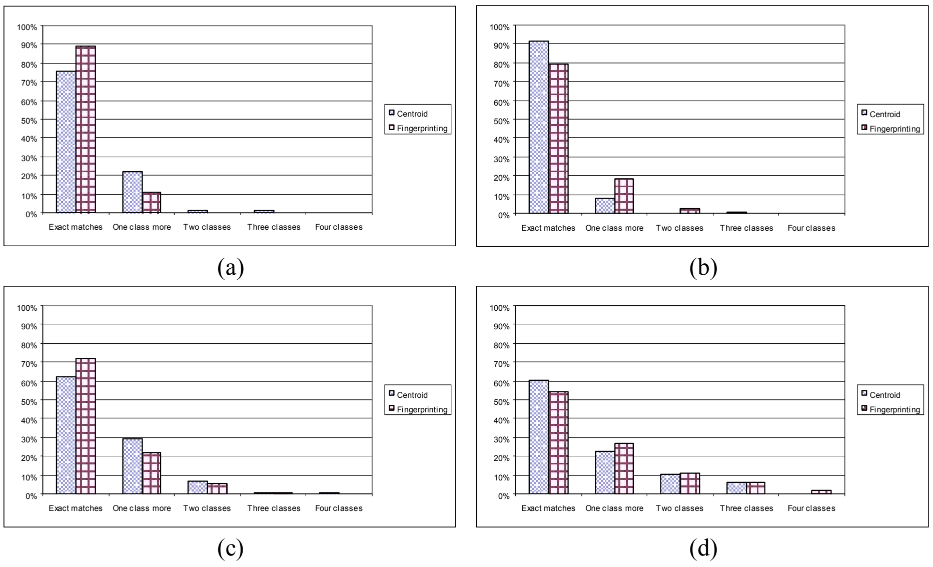
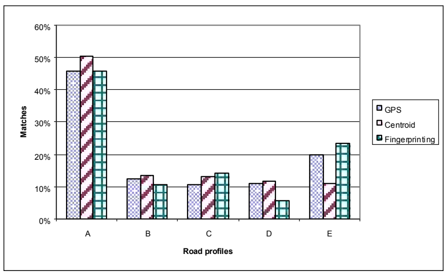


Fig. 9. Speed profile matches for GPS and WPS derived travel times, for various sampling periods: (a) 5sec, (b) 10sec, (c) 20sec, and (d) 30sec.

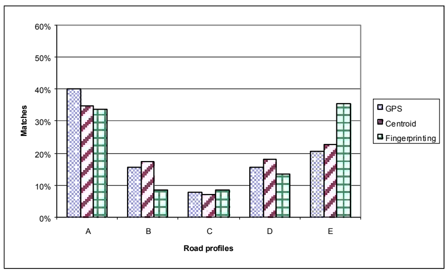
We experimented with such quantization in travel time speeds and introduced for our experiments five road categories characterized by the following speeds (in kph): [0-10), [0-20), [20-30), [40-50), [50,∞). We classified all road links based on GPS and WPS data, and for various sampling frequencies. Further, for each road link in our network, we compared the classification produced from GPS, WPS-C, and WPS-F.

Our results are shown in Fig. 9. For example, in Fig. 9(a), 75% of the road links are classified under the same category for WPS-C, compared to GPS. For WPS-F, this number is close to 90%. From Fig.9, we can also observe that for sampling rates of 5s and 10s, at least one of the two WPS techniques derives the same road categories for 90% of the road links. As the sampling rate decreases, this percentage is reduced to roughly 60%, with additional 25% of the road links classified to one category higher or lower. Therefore, we can conclude that for higher sampling rates, WPS produces very accurate travel times which are indeed comparable to GPS. For lower sampling rates (30s) the results are encouraging, since at least 80% of the derived travel times fall within the same or a directly neighboring category.

What follows in Fig.10 is the *actual link classification* based on GPS and WPS. Fig.10 shows the percentage of road links that fall in one of our five categories for GPS, WPS-C and WPS-F. It is evident that for small and high sampling rates alike, a WPS derived classification is very similar to a GPS classification.



(a)



(b)

Fig. 10. Road segment classification for (a) 5sec and (b) 30sec

5 Conclusions and Future Work

We have evaluated the use of WPS data as an alternative data source for extracting travel times for road networks. We adapted and evaluated various classes of the centroid and fingerprinting WPS algorithms. Further, we applied map matching as a post processing filter to improve WPS accuracy and demonstrating significant gains. In addition, we extracted travel times from GPS and WPS data with a map-matching algorithm. Our evaluation demonstrated that for measurement periods up to 10sec, the produced travel times are practically identical to the ones derived from GPS data. Further, when applying a typical speed profile classification on travel times, even for sampling rates of up to 30sec, the produced travel times are still of respectable quality. Finally, we showed that through our analysis of WPS data, the distribution of road segments to speed profiles can be accurately discovered.

Our ongoing work evolves around further exploring and manifesting the benefit and potential uses of huge amounts of crowd-sourced WPS data. In this respect, our efforts are focused on three fronts. First, improve the accuracy of WPS techniques by integrating map matching into the WPS algorithms. Second, explore different uses for WPS data, such as routing (by fully replacing GPS), and automatic road network construction. Third, we aim to model and accommodate the inherent inaccuracy of wireless positioning data sources into spatiotemporal tasks and algorithms.

Acknowledgements

This work was partially supported by the project "*TALOS: Task Aware Location Based Services for Mobile Environments*", funded by the European Community, Framework Programme 7, Research for the benefit of SMEs. We would like to thank Dimitris Sacharidis, Kostas Patroumpas, Theodore Dalamagas, and Panagiotis Bouros for their valuable comments.

References

1. Alt, H., Godau M.: Computing the Fréchet distance between two polygonal curves. Int. J. Comput. Geom. Appl. 5, 75–91 (1995)
2. Bahl P., Padmanabhan V.N.: RADAR: An In-Building RF-Based User Location and Tracking System. In : 9th IEEE Conference on Computer Communications, pp. 775-784, IEEE Press (2000)
3. Bahl, P., Padmanabhan, V.N., Balachandran, A.: Enhancements to the Radar User Location and Tracking System. Technical Report, Microsoft Research MSR-TR-00-12 (2000)
4. Brakatsoulas, S., Pfoser, D., Salas, R., Wenk, C.: On map-matching vehicle tracking data. In: 31st Very Large Data Bases Conference, pp. 853–864, ACM (2005)
5. Brockfeld, E., Wagner, P., Passfeld, B.: Validating travel times calculated on the basis of Taxi Floating Car Data with test drives. In: 14th World Congress on Intelligent Transport Systems (2007)
6. Cheng, Y., Chawathe, Y., LaMarca, A., Krumm, J.: Accuracy Characterization for Metropolitan-scale Wi-Fi Localization. In: 3rd International Conference on Mobile Systems, Applications, and Services, pp. 233–245, ACM(2005)
7. Chen, M.Y., Sohn, T., Chmelev, D., Hightower, D.H.J., Hughes, J., LaMarca, A., Potter, F., Smith, I., Varshavsky, A.: Practical metropolitan-scale positioning for GSM phones. In: 8th International Conference on Ubiquitous Computing. LNCS, vol. 4206, pp. 225-242, Springer (2006)
8. Hightower, J., Consolvo, S., LaMarca, A., Smith, I., Hughes, J.: Learning and recognizing the places we go. In: 7th International Conference on Ubiquitous Computing. LNCS, vol. 3660, pp. 105-122, Springer (2005)
9. KISMET, http://www.kismetwireless.net/
10. Krishnan, P., Krishnakumar, A.S., Ju, W., Mallows, C., Ganu, S.: A system for LEASE: Location estimation assisted by stationary emitters for indoor RF wireless network. In: 23rd IEEE Conference on Computer Communications, pp. 1001-1011, (2004)
11. LaMarca, A., Chawathe, Y., Consolvo, S., Hightower, J., Smith, I., Scott, J., Sohn, T., Howard, J., Hughes, J., Potter, F., Tabert. J., Powledge. P., Borriello. G.,

Schilit. B.: Place Lab: Device Positioning Using Radio Beacons in the wild. In: 3rd International Conference on Pervasive Computing. LNCS, vol. 3468, pp. 116-133, Springer (2005)

12. Laitinen, H., Lahteenmaki. J., Nordstrom. T.: Database correlation method for GSM location. In Proceedings of the 53rd IEEE Vehicular Technology Conference, pp. 2504-2508, IEEE Press (2001)
13. Otsason, V., Varshavsky, A., LaMarca, A., Lara, E.D.: Accurate GSM Indoor Localization, in Proceedings of Ubicomp. LNCS, vol. 3660, pp. 141-158, Springer (2005)
14. Pfoser, D., Jensen, C.S.: Capturing the Uncertainty of Moving-Object Representations. In: 6th International Symposium on Advances in Spatial Databases. LNCS, vol. 1651, pp. 111-132. Springer (1999)
15. Quantum GIS Project, http://www.qgis.org/
16. Schaefer, R.P., Thiessenhusen, K.U., Wagner, P.: A Traffic Information System by Means of Real-time Floating-car Data. In : 9th World Congress on Intelligent Transport Systems (2002)
17. Sohn, T., Varshavsky, A., LaMarca, A., Chen, M.Y., Choudhury, T., Smith, I., Consolvo, S., Hightower, J., Griswold, W.G., Lara, E.D.: Mobility Detection Using Everyday GSM Traces. In: 8th International Conference on Ubiquitous Computing. LNCS, vol. 4206, pp. 212-224, Springer (2006)
18. Varshavsky, A., Chen, M., Lara, E.D., Froehlich, J., Haehnel, D., Hightower, J., LaMarca, A., Potter, F., Sohn, T., Tang, K., Smith, I.: Are GSM phones THE solution for localization?. In: 7th IEEE Workshop on Mobile Computing Systems and Applications, pp. 20-28, IEEE Press (2006)
19. Wenk, C., Salas, R., Pfoser, D.: Addressing the Need for Map-Matching Speed: Localizing Global Curve-Matching Algorithms. In: 19th Scientific and Statistical Database Management Conference, pp. 379-388 (2006)
20. European Space Agensy - Galileo, http://www.esa.int/esaNA/galileo.html
21. Hellenic Positioning System, http://www.hepos.gr/