OLBS: Offline Location Based Services

P. Coelho[1], A. Aguiar[2], J. C. Lopes[3]
[1]Fraunhofer Portugal AICOS, [2]IT Porto/FEUP, [3]INESC/FEUP
[1]pedro.coelho@fraunhofer.pt, [2]ana.aguiar@fe.up.pt, [3]jlopes@fe.up.pt

***Abstract*—Most existing location-based services rely on ubiquitous connectivity to deliver location-based contents to the users. However, connectivity is not available anywhere at anytime even in urban centres. Underground, indoors, remote areas, and foreign countries are examples situations where users commonly do not have guaranteed connectivity but could profit from location-based contents. In this work, we propose an open platform for publishing, distributing and maintaining location-based contents that can be accessed offline by the user on a mobile device. Additionally, we describe the prototype we implemented for proof of concept consisting of an smartphone application and a web application backoffice, using 2D-barcodes as location identifiers. Our prototype provides offline access, as well as publishing, installing and updating location-based content.**

***Index Terms*—content delivery; location-based services; mobile applications; Web-based application; Android OS**

I. Introduction

Mobile phones have evolved into devices with multiple connectivity (3G, bluetooth, WiFi) and capable of a lot more than just making phone calls, fostering the development of a wide range of enhanced applications that rely on ubiquitous connectivity. Location-based services (LBS) are very popular among them, offering on-demand navigation like GoogleMaps [1] or GoogleTransit [2], identification of nearby restaurants or other points of interest like Place Directory [3], reviews about nearby places, like Yelp [4], leaving virtual notes, sharing and viewing location of people in a social network like Foursquare [5] or Gowalla [6], among many others.

However, connectivity is far from being ubiquitous and it will be long before it is so for everyone due to a wide range of reasons, raging from actual lack of wireless connectivity to lack of financial capability to access it. The first case is true for several areas outside of the main urban centres where cellular data services are not available, or underground and indoors areas, where cellular coverage is poor and WiFi access points are not freely accessible. Moreover, not all mobile devices that might be used for LBS are equipped with both cellular and WLAN connectivity, e.g. the iPod Touch. On the other hand, while the high costs that keep ubiquitous wireless access from being universal tend to disappear within national borders, but not outside them. Although tourists/travellers configure a main target group for LBS, the need to be online associates prohibitive costs with the use of such applications for many users.

Navigation and tourist guide applications have recently shown a tendency to offer offline access to contents. The downturn of these solutions is that they provide the contents in a proprietary format embedded into applications. Even when user-produced data sharing is built-in, it is only possible within each application, as in Gowalla or Foursquare. This state of affairs forces the user to download, get acquainted with and use a multitude of different applications for different purposes, geographical areas and types of content. A cachable version of GMaps is still one of the most requested items in the category Location-Based Services of Google Product Ideas[1].

This work proposes an open platform for distributing and maintaining packages of location-based content that can be accessed offline on a mobile device. We envision a platform that enables access to contents associated with physical places without requiring permanent Internet connectivity. Moreover, the platform should support various types of location sensors and enable new ones to be easily added. Finally, the platform should be open, clearly specifying a format for location-based content packages, as well as interfaces between different entities in the platform. Here, we make several contributions related to the data structures and interfaces necessary for an open platform. 1) We propose an open and uniform way to describe the association of contents with a location, supporting different types of content formats and different types of location sensors. 2) We specify an open platform that enables offline access to LBS, namely

[1]The website is being re-directed by Google, but it is still accessible if you enter the following url by hand: http://productideas.appspot.com//#15/e=cf&t=2d8d1

displaying information about a location or displaying the location on a map together with nearby points of interest. We do not consider navigation at this stage, as it would impose a high processing burden on the resource limited mobile device. 3) We propose a solution to enable intuitive management (download, update, deletion) of location packages.

The remainder of this paper is organised as follows: the next section briefly surveys related work, Section III describes a uniform representation of location-based information, Section IV describes the functionality of the platform and Section V specifies the interfaces, Section VI describes the proof of concept prototype, and Section VIII discusses future work and concludes the paper.

II. Related work

We divided related work in two main fields: architectures and models for context-aware services, of which location-based services are a specific case as described by Abowd et al [7], and existing applications that provide location-based services. A large amount of effort has been dedicated to designing architectures, frameworks and models for context-aware services, and we refer to [8] for a representative survey and a taxonomy for their classification. We opted for a middleware application architecture, which separates application logic from low level components, providing scalability and re-usability to the system, as well as hiding low level components from the application developer.

A simple hierachical architecture, as proposed in [8], offers too little flexibility, since we aim at supporting multiple location sensors, like GPS, RFID [9] or 2D-barcodes [10]. The Context Toolkit [11] architecture resembles a peer-to-peer network architecture, where a widget is associated with a sensor unit, has a particular state depending on what context information it is monitoring and can be queried by applications. Applications can register to be notified of context changes detected by a widget through a callback. Composite widgets may also be used to combine the state of other widgets. The Context Managing Framework [12] proposes an architecture that provides distributed components for the low level sensors (resource servers), which are unified by the context manager acting like a blackboard. The context manager then acts as a server to applications. The context manager, any resource servers, and applications run on the mobile device itself. Our application borrows concepts from these two architectures.

Among the wide range of recently emerged LBS applications, we mention here only a few examples related to our work. GoogleMaps [1] allows users to view their location on a map, shows nearby contents added by other users and offers navigation services. Place Directory [3] identifies nearby restaurants or other points of interest, Yelp [4] presents reviews about nearby places, Foursquare [5] or Gowalla [6] enable sharing and viewing location of people in a social network, together with other functionalities, like leaving virtual notes. These applications, as many others of the kind, rely on connectivity at the time the users access the services but do not allow users to take a snapshot of the data with them for later offline access.

Signpost [10] is an application developed to offer indoor navigation using camera-enabled devices using visual markers (2D-barcodes) for indoor location sensing, and does not rely on connectivity for displaying the user position on a map and showing nearby points of interest. However, it uses a closed data format, which requires that the map and location information be compiled into a standalone application that offers the service offline in the form of a closed and proprietary application. Recently, enhanced user interaction and localisation techniques for location-based services were proposed [13], but no considerations are made regarding the lack of ubiquitous connectvity.

The platform that we present in this work differs from these applications in that it does not requires connectivity to offer location-based services. We specify a data format and a communication model between platform components, so that anyone can create packages of information and publish them through the platform, and multiple information package servers and reading applications can co-exist and interact.

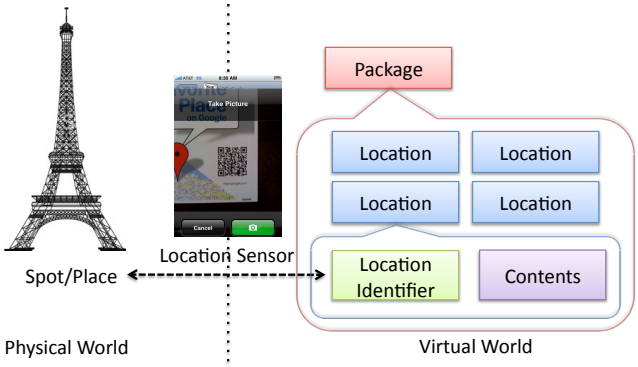
III. Location, Location Identifier and Package

In this section, we define the information structure used to represent places and associated contents.

A *Location* is a representation of a spot/place in the physical world. It exists in the virtual world, must be associated with at least one location identifier and must be unique within a *package*. Each location can be associated with various types of content, where *Content* is information stored in a file in one of several supported formats. A *Location Identifier* is the output of a location sensor and identifies a spot/place in the virtual world. It is the key that binds the spot/place to a location in the physical world with the location in the virtual world.

The granularity of the place that the location identifier refers to may vary, ranging from geographic coordinates, through name of a monument/building, to a room inside a building or even a static object, like a vending machine or an exhibit in a museum. A *Location Sensor* is a device that associates a spot/place in the physical world with a location identifier in the virtual world. A *Package* is an organisational entity in the platform that aggregates a collection of *Locations* and their associated *Contents*. These concepts are depicted in Figure 1.

Fig. 1. Overview of the information structure about a place in the physical world



IV. OLBS: Offline Location-Based Services Platform

In this section, we give an overview of the platform as a whole and describe its functionalities.

A. Platform Overview

We propose a mobile application that communicates with repositories for retrieving content packages following a client-server model, and a specification of the data formats used to implement the information structure defined in the previous section. The envisioned use scenario is that a user choses and downloads content packages from a repository where they were published, using the mobile application when he has connectivity. The contents are stored in the smartphone and the user can access them later on offline. Additionally, the user is notified of content package changes so that he can update his local version. This last functionality is critical to provide comfortable use of offline versions of content packages that result from collaboration or that are regularly improved. A possible use case could be the content package for the temporary exhibition of a museum, or a snapshot of GoogleMaps or Yelp.

B. Content Distribution and Management

Content packages must be distributed prior to their use, remaining stored in the user's device and being updated when connectivity is available upon explicit user request. We propose an approach similar to existing Linux package distribution systems. A package management system is a collection of tools to automate the process of installing, upgrading, configuring, and removing software packages from a computer. So, a user pre-installs a package from a repository, and the content associated with the locations in that package is then available offline.

A package may suffer updates, additions or deletions, and new packages may be added to the system. Therefore, the platform includes a package manager that manages package transactions. The package manager client is part of the application and the server lies in a remote backoffice, co-located with the package repository containing available location-based content packages. The application connects to the repository using the package manager to search, add or update packages. Packet deletion from the mobile application must currently be explicitly requested by the user.

C. Offline Access

To enable the user to access location-based contents while offline, the content packages are stored in the mobile device. Only the organisation and indexation of contents is done inside the application, while the contents themselves are stored as files in the file system.

Access to location-based contents is triggered by the user from inside the application. I. e., the user controls the process by requesting readings from location sensors (e.g. a 2D barcode or RFID tag), which respond with a location trigger to the location manager in the application. Upon reception of this trigger, the application accesses the local database to retrieve the path to the contents associated with that location, then retrieves and displays them.

D. Content Publishing

Packages can be prepared by anyone who wishes to share location-based information for a set of locations, as long as they follow the specified format. Users who produce location-based information will from now on be called publishers. A publisher user management and authentication system is required, and eventually a reputation system may be added, but those matters are out of the scope of this work.

V. Specification

A. Information Structure

An information package has the attributes name, version and description, the latter being useful when the user is searching for packets. A location has the attributes name, location identifier and tags. A location identifier has a single attribute, which is the identifier itself. This is how a location may be retrieved using not only one but several identifiers, which can occur in different formats or sources: data embedded in a 2D barcode, positioning data retrieved from GPS or other sensors. Each item of content associated with a location has a name, a path pointing to the corresponding file in the file system, zero or more tags to characterize the content and its MIME-type, in order for the application to know how to handle it. These structures can be visualised in Figure 3, in Section VI where we show the structure of the databases used in the prototype.

B. Application-Repository Interaction

All communication between application and repository uses HTTP, as the primitives and the request-response transactions provide all the functionality required and the stability and robustness of its network stack are well proven. The backoffice implements a RESTful webservice that exposes the *package*, *location* and *content* resources. Therefore, a GET request on either one of these three resource URLs (e.g. `GET /packages`) returns the collection of the respective entities, whereas a request directed towards a specific entity identified by a sequential integer (e.g. `GET /packages/2`) returns the meta-data of that specific resource. Similarly, other HTTP verbs may be used to execute other actions beyond fetching (e.g. using POST to create new entities). The integer specifying an entity can be retrieved from the response to GET on the aggregating resource, i.e. packages for location identifiers, and locations for contents identifiers. This is relevant to enable updating single resources within a package.

A return format may also be specified by appending its standard file extension to the URL (e.g. `GET /locations.xml`, or `GET /packages/2.json`). If no format is specified, an HTML page describing the requested entity is returned. Included formats are XML and JSON.

Between JSON and XML, the first is preferred because it is light in size and complexity without sacrificing portability. Though XML may be more descriptive, it is also more verbose and more complex. Since its verbosity is of no use in this particular case, the extra amount of data to be transferred would present no benefit.

As a repository contains files with the actual contents in a binary format besides metadata, these are compressed in the ZIP format and fetched when the user downloads a package after fetching its meta-data. Figure 2 shows an example of a JSON response when querying a repository for a *package* listing (`GET /packages.json`).

Fig. 2. Example of a package listing response from the repository

```
[
  {
    "package": {
      "name": "FraunhoferAICOS",
      "updated_at": "2010-06-01T18:23:02Z",
      "id": 3,
      "version": 6
    }
  },
  {
    "package": {
      "name": "FEUP",
      "updated_at": "2010-05-25T19:25:15Z",
      "id": 4,
      "version": 2
    }
  }
]
```

VI. Prototype

As a proof of concept and verification of the specification, we implemented a prototype comprising of a mobile application for the Android Operating System and a Web application, containing a repository for content publishing. Besides verification that the proposed platform provides the envisioned functionality, the purpose of implementing and using the prototype was to identify additional features that might be useful or even required, for management or usability purposes. The Android application was set up in an HTC Magic smartphone, running the Android OS 1.5.

A. Location Sensors

We used 2D-barcodes as a location sensor, similarly to the Signpost [10] application. We chose QR-Codes since they have proven to be the most reliable and legible in tests performed by Kato et al. [14].

B. Backoffice

The backoffice serves as a package repository, implementing a package versioning strategy. Additionally, to

ease package building, it also functions as a package generator and publisher, converting locations and associated contents into a package format that the application can download and read. A publishing interface allows a publisher to create new packages, add locations to those packages, edit their meta-data, and add additional contents to the locations created. Upon generating a location, the interface exposes a generated QR Code containing an identifier to the location itself. That QR Code will later be used as the visual tag that identifies a location, enabling the mobile application to access the contents associated with that location. Published packages may receive additional contents or locations, or have them removed without affecting the currently published version of that package. Changes between versions are also tracked, enabling the publisher to track when a certain content was added, removed or modified.

Having the platform generate the packages also enables future optimisations, like adapting the formats of contents to mobile device capabilities. We identify that process as "content adaptation".

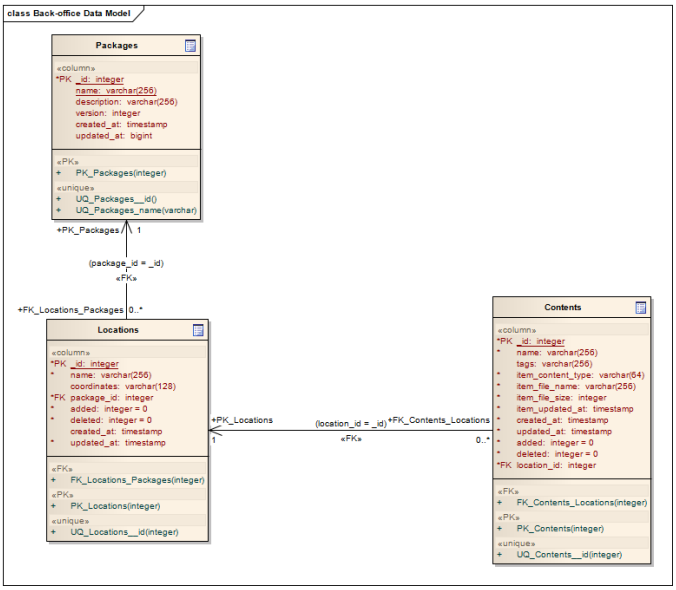
C. Mobile Application

The mobile application prototype component implements listing, retrieval and update of packages published on a single backoffice, as well as location-based content retrieval, caching and access. It connects to the backoffice in order to list, download and install packages or their updates. The remainder of its operation is done entirely off-line. Access to contents associated to a location is done in a simple way: by scanning the QR Code that marks a location.

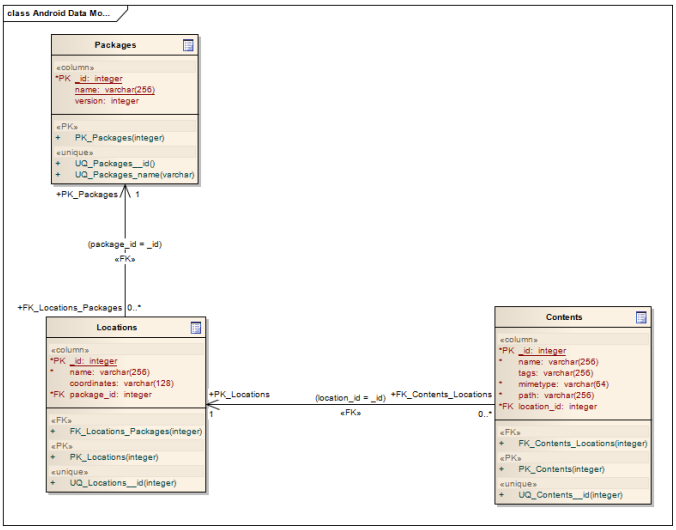
D. Packages of Location-based Information

Each package contains locations with associated meta-data and content stored in files, as specified in Section V. The list of locations and their meta-data is stored in a database, together with links to content files in the file system. This is so both in the repository and in the application, whereby PostgreSQL was used in the first because it has a decent fulltext search implementation out of the box, and SQLite3 was used in the application, as it is the SQL engine that bundles with Android, featuring also decent capabilities. Figure 3 shows the structure of both databases, where some differences can be seen with respect to the specification that proved relevant to the deployment, like the tags to classify contents or the timestamps required for the packet versioning in the repository.

Fig. 3. Database structure for the package repository and application



(a) Repository



(b) Application

VII. A Prototype Package

We generated a package for the premises of Fraunhofer Portugal AICOS, at Campo Alegre, Porto, which displays information associated with a location. The package includes a set of locations that describe a room or a desk. Each "room" location is associated with various HTML files that describe the room itself and the people who work there. "Desk" locations describe textually via an HTML file the person that works there, attaching a picture of that person. These locations also describe the projects that the person positioned there works on, through text, pictures and sometimes video

or sound. The test package contained 17 locations, each with 1 to 9 associated content files of type HTML, JPG or PNG. The package contained 9 PNG files, 23 JPG, 29 HTML and 1 GIF, adding up to a package file size of 6,403,947 Bytes[2], an acceptable value for state-of-the-art mobile devices.

The package was built entirely using the back-office, and installed or updated using the mobile application. Locations can be added or removed, meta-data edited and published, creating a new version of the package, which triggers an update notification when the mobile device is within the range of an WLAN and the mobile application is started. All these interactions works as planned, thus making the prototype fully functional.

The application was successfully deployed in the scenario depicted above. It performs as it should, displaying off-line content triggered by a QR Code scan positioned at the physical location. The mobile application currently occupies 512 kB in the smartphone's internal memory, while the relational database fits in an extra 8 kB, providing it stores only the meta-information of the test package. The total sum is 520 kB occupied in the smartphone's internal storage for the application and the package meta-data and roughly 6.5 megabytes in the external storage, in this case an SD card, for the content files.

VIII. Conclusions

We propose Offline Location Based Services an initial step towards an open platform for publishing, distributing and maintaining location-based contents that can be accessed offline by the user on a mobile device. Location-based contents are organised into packages that can be downloaded and stored on the mobile device for offline use.The platform consists of a backoffice that has a repository for location packages, which are managed in a way similar to software package managers, providing a friendly way for the user to list, download, update and remove packages from his device. Additionally, the platform provides a de-coupling between location sensors and information associated with a location, enabling the support of more than one location sensing technology. The platform is designed to provide two common functionalities associated with location-based services: delivering information associated with a location and pinpointing the user's location on a map.

We implemented a prototype of the platform on a state-of-the-art Android smartphone using 2D-barcodes as visual marker for proof of concept. We verified that the prototype works as expected with respect to package generation, publishing, editing, updating, and accessing the contents offline. The prototype backoffice and application are published under the LGPL license and can be downloaded from the following locations: http://github.com/punnie/OpenLBS-Backoffice and http://github.com/punnie/OpenLBS-Android.

Currently, the prototype only implements viewing contents associated with a location, and the next step is the implementation of the pinpointing functionality, followed by content adaptation. Another open matter is the design of an algorithm to suggest packet deletions in the mobile application. Further work also includes extending the platform to enable users to add information to a location offline and synchronising it at a later time. This will enable a more powerful interaction between the users and the environment.

[2]The package size is the sum of the file sizes, since no content adaptation is implemented at this stage.

References

[1] Google. Google maps api. [Online]. Available: http://code.google.com/apis/maps/

[2] ——. Google maps transit. [Online]. Available: http://maps.google.com/help/maps/transit/

[3] ——. Place directory. [Online]. Available: http://googlemobile.blogspot.com/2009/06/places-directory-app-for-android.html

[4] Yelp. [Online]. Available: http://www.yelp.com/

[5] Foursquare. [Online]. Available: http://foursquare.com/

[6] Gowalla. [Online]. Available: http://gowalla.com

[7] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a better understanding of context and context-awareness," in *HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*. London UK: Springer-Verlag, 1999, pp. 304–307.

[8] M. Baldauf, S. Dustdar, and F. Rosenberg, "A survey on context-aware systems," *Int. J. Ad Hoc Ubiquitous Comput.*, vol. 2, no. 4, pp. 263–277, 2007.

[9] H. D. Chon, S. Jun, H. Jung, and S. W. An, "Using rfid for accurate positioning," *Journal of Global Positioning Systems*, vol. 3, no. 1-2, pp. 32–39, 2004.

[10] A. Mulloni, D. Wagner, I. Barakonyi, and D. Schmalstieg, "Indoor positioning and navigation with camera phones," *IEEE Pervasive Computing*, vol. 8, no. 2, pp. 22–31, 2009.

[11] D. Salber, A. K. Dey, and G. D. Abowd, "The context toolkit: aiding the development of context-enabled applications," in *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA: ACM, 1999, pp. 434–441.

[12] P. Korpipaa, J. Mantyjarvi, J. Kela, H. Keranen, and E.-J. Malm, "Managing context information in mobile devices," *IEEE Pervasive Computing*, vol. 2, no. 3, pp. 42–51, 2003.

[13] D. McGookin, C. Magnusson, M. Anastassova, W. Heuten, A. Renteria, and S. Boll, Eds., *Proc. of Workshop on Multimodal Location Based Techniques for Extreme Navigation at Pervasive 2010*. FP7 Project "HaptiMap", May 2010.

[14] H. Kato and K. T. Tan, "Pervasive 2d barcodes for camera phone applications," *IEEE Pervasive Computing*, vol. 6, pp. 76–85, 2007.