A mobile cloud service for traffic lights with audible waring localization

Alberto Franco *University of Padua*, Marco Sbrignadello *University of Padua*

***Abstract*—The aim of the project is to create an application that automatically localize traffic light with audible warnings. After the tracking has been done the idea is to generate directions on map applications (such as Google maps) for visually impaired people. The software system in analysis is made by a mobile application that gather the data and a cloud service that manage it. This topic is of particular interest (as stated in [1]) and we take the challenge to understand what is the best solution to carry out the task. In this paper we explain how we did implement a solution for the problem and the possible improvement that can be done to it and which limits has our solution.**

***Index Terms*—cloud, service, wireless, network, traffic, light, localization, android, google, app, engine, mobile, development, DTN**

I. Introduction

"Technology is the usage and knowledge of tools, techniques, crafts, systems or methods of organization in order to solve a problem or serve some purpose" (cit. Wikipedia). We should use technology to help people with disabilities in order to simplify their lives. We thought that visually impaired people may be misguided by some indications give them by map applications that does not consider their disability. The first step to fill the gap is to gather the data to generate correct directions depending on the disability the person holds.

The problem. For visually impaired people the main problem is to cross the road in a safe way, to help them local governments has endowed traffic lights with audible warnings (TLAW from now on for sake of brevity). These sounds are produced when the traffic light turns green and they are not while the light stay red. With this idea in mind we thought to localize them with an automatic system that everyone can use to serve the cause. With the modern smart-phones and PDA holding lots of technology the task is simplified and the data can be gathered.

Our solution. The easiest way we found to solve the problem has been to create a mobile application (we developed it for Google Android) that communicates with a cloud service[1] (developed with Google App Engine). The data is gathered through a recognition of walk-stop sequence in the mobile device, every time the carrier of the device stops the application change state and get ready to record data, once the person start again to walk the device records ten seconds of microphone capture and append it in a queue of transfer to be done. We decided to use a DTN[2] approach to send data, once there is an available connection to the Internet the application send the gathered data to the cloud service that analyze it, in order to understand if it is a valid recording of a TLAW or not. Once decided that the cloud service store the data only if it is valid. An other way to solve the problem would be through image analysis as done in [3] but this would require active participation of the user that, every time he finds a TLAW, should make a photo of the traffic light and still we are not sure that the traffic light has audible warning so it would be more complicated to track TLAW position.

II. The Cloud Service

The component of the software systems that manage the data has been developed as a cloud service. We thought to develop the data analysis and management system as a cloud service in order to have a scalable, extensible and distributed environment. After trying few options as development environment for cloud computing (we actually test Google App Engine[5] and Windows Azure) we decided to use Google App Engine for the availability of examples and the policy of Google respect to open source and academic projects. In order to use Windows Azure we should pay and was not worth for such a small project, despite its capability are superior to Google App Engine (e.g. high throughput with CDN). We do not developed the signal analysis component because we are not competent enough to develop such a piece of signal analysis software.

The service. Everything works in a simple way. Every time a transmission is received the data is analyzed (in

[1]With cloud service we means a distributed application running on demand over a large number of servers.

[2]Delay Tolerant Network

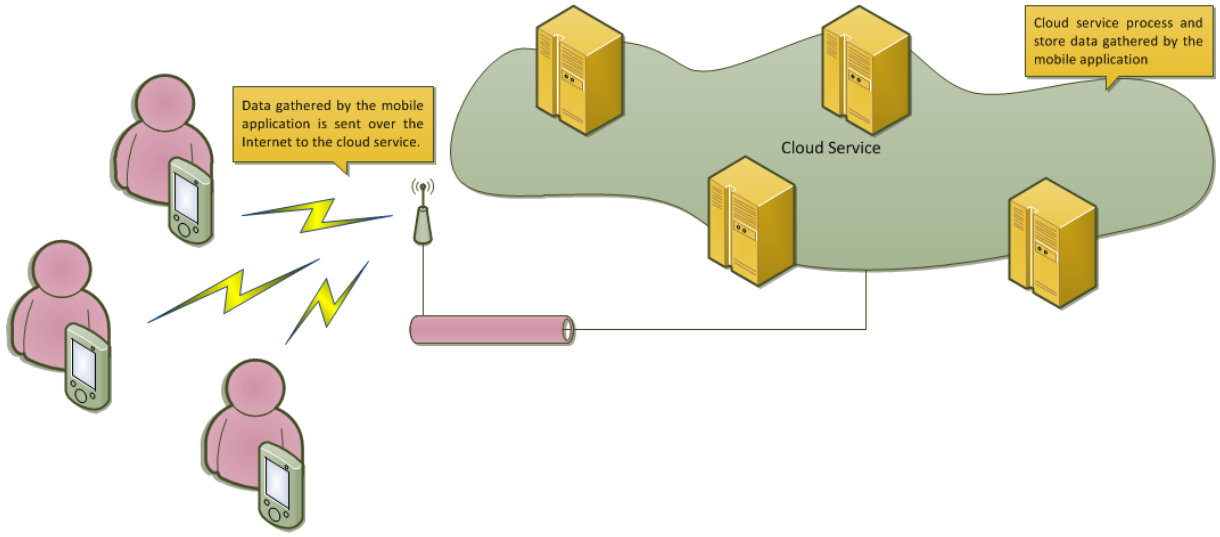

Fig. 1. System Architecture

theory, not in our implementation) and stored. If the record sent to the server contains an audible warning like the ones used in the traffic lights to help visually impaired people the server keeps the data else the data is discarded. The data packet contains:

- The recording done by the user on-place.
- Coordinates in longitude and latitude (get with GPS).
- A time stamp to identify the recording.

With this data we can identify in a unique way the recording. Follows the transfer sequence description: the file is serialized and sent through a *http post request*, the service answer with an OK message and meanwhile calculate the MD5 sum of the data. The application has a verification of the transfer case and can send the additional data (in a *http get request*) such as time stamp, longitude and latitude with the MD5 sum calculated on the data sent. This should be the moment where the data is analyzed and eventually discarded.

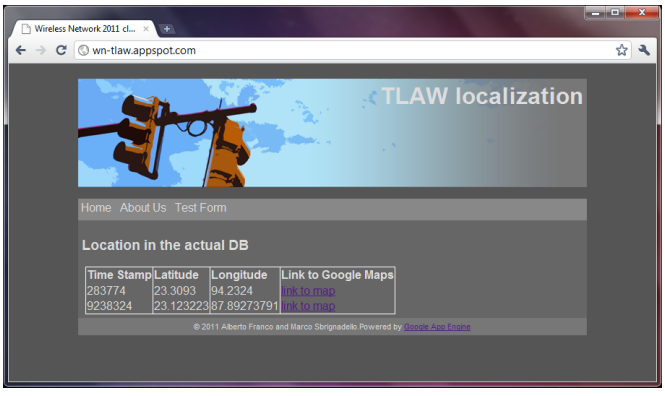

Fig. 2. Screenshot of the web interface

Performance of the service. Having a sufficiently wide bandwidth the system runs well. The amount of data to send per each requests is little (in the order of 10-20 KB) so the use of such protocol to send data does not slows down the overall performance of the software. Using a cloud service we have potentially unlimited calculation power (we are limited only by the free quotas given by Google) and the elaboration of data should be fairly fast. The use of other methods such as REST or javascript sockets would be more efficient but we decided to use this type of protocol for it is supported by every platform and extending the software system would be easy.

III. The Mobile Application

The component of our solution that aim to record the audio and take care of upload it to the server has been developed as a mobile application. We decided to use Google Android[6] operating system for the development because it provides all the tool to make a testing application quickly and also because it would be easy to extend our software in the future. We imaged an utopian scenario where every smart-phone runs in background our application. The idea is to have a daemon that take care of analyzing users movement on the street, records the audio at the appropriate time and, after acquiring geographic coordinates, send them to the cloud service.

The application. We thought the mobile application divided in three parts:

- the upload manager;
- the recording system;
- the sensor manager.

Upload manager. We decided to develop the upload manager of the application first because of its

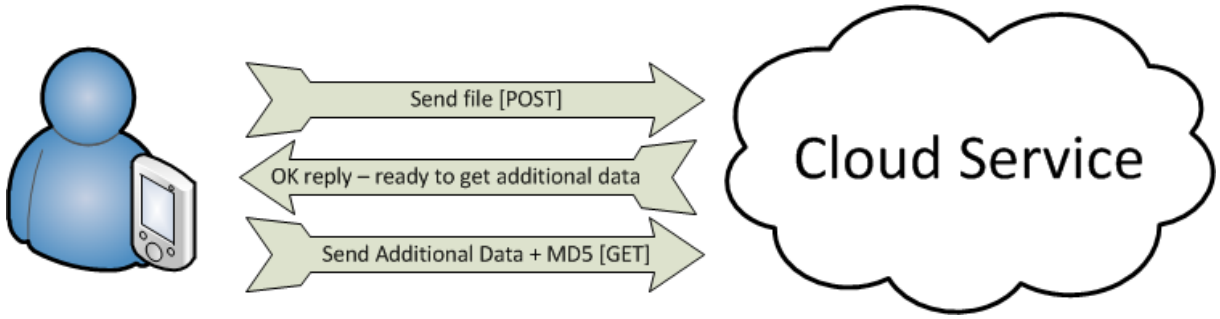

Fig. 3. Transfer Sequence

complexity; indeed in this part we implemented the client-side communication protocol and the smart-phone connectivity manager. Our application is thought to work in a delay tolerant fashion; the recording system saves the audio file locally in the device and, only when an Internet connection is available, the upload manager starts the upload. Then the upload manager has been implemented as a thread and its tests periodically the Internet connectivity and, when it is available, upload any file stored locally. When the thread is started (only when there are audio file) it checks the connectivity status of the device and, if it is not available, it set itself to sleep for one minute using a countdown timer provided by the operating system. In this way it does not increase the power consumption of the application by doing useless operations. We thought that the application does not need to be notified in real time when the Internet connection become available, so the connectivity check is done every minute without affecting system performances. After that, it connects to the cloud service and starts uploading the file following the protocol explained above. When the upload manager finishes the upload it removes the local file and it controls if there are other files to send; if there are no files to send the thread is set to sleep, otherwise it continues to upload.

Recording system. The recording system is started by the sensor manager (when needed) and starts to record an audio file using the integrated microphone of the device. The portability of this operation is possible thanks to the Google Android Framework that allows to use correctly every integrated hardware of all Android devices. We decided to make audio recording lasting 10 seconds for two reason: the audio retrieved lasts enough to allow the cloud service application to detect if there is an audible warning in the recording; the audio file is not great then 15 KB and so can be uploaded quickly. The recording use the encoding .3gp[3] provided by the operating system; this is a lossy audio compression but it provides a good quality, we are not sure if the analysis will be correct. Another task of the recording system is to store the audio file locally in a simple database and associate it the geographic coordinates retrieved by the sensor manager when the recording starts.

Sensor manager. The sensor manager starts when the application is launched. It manages all the sensors used by the application (accelerometer and GPS). First of all it checks GPS status: it notifies the user to enable the GPS connectivity if it is turned off and than close the whole application; if the the GPS connectivity is turned on, it tries to connect to the satellites and only if the device is connected, it launches the accelerometer manager. While the GPS manager continuously tracks geographical location, the accelerometer manager analyze users movement. While the user walks the accelerometer detects variations and continues to analyze it. When user stops there would be no variations in the accelerometer data and than it means the user could be in proximity of a lighter. When he starts to walk again the accelerometer manager detects this movement and than starts the recording, this is done because the audible warning are produced when the traffic light is green so people can cross the road. If the user does not cross the road with the red light our system is able to detect the stop-walk sequence and correctly record the data. Some of the data gathered will be inconsistent thats why the cloud service must analyze all the data.

IV. Conclusion

Using this software system to track TLAW is a really interesting idea but has some glitches. What we have experimented is that microphone recording with the phone in pocket does not give a good quality sound. Our signal processing experience is very little so we cannot say if the resulting recording may be analyzed in order to understand if it is usable or not. The pattern of audible warning itself is unknown, we suppose that all traffic light in the world has the same type of audible warning.

[3]Actually 3gp is the containing format, the audio compression format is AMR (Adaptive Multi-Rate).

We use a little potential from Google App Engine, you may think to extend the use of the cloud service to relieve the mobile device of some task. Indeed we have designed a good service that can be improved but has some good points, the overall logic is there and the only part that is missing is signal analysis module.

V. Future Works

The major improvement that can be done in the software system is writing a good protocol to record the data. Actually we pass through *http requests* that are not that fast as we would like. The best would be using sockets and transfer the data using all the bandwitdh TCP delivers to application layer, the use of http requests limits the performance of the software system, more reliable and fast methods to store data are found in [2], the implementation can be improved.

Major improvement can be done also in the mobile application, the management of sensors is one of the hot points of the app. A better control over GPS (or totally substitute it with cell triangulation) should be done and the use of accelerometer to understand user movement should be implemented. Also some performance issues and power consumption optimization should be done.

References

[1] Bhargava, Pelin, and Lian Duan. *A Mobile-Cloud Pedestrian Crossing Guide for the Blind.* ICACC-2011, 2011.

[2] Bogdan Nicolae. *High Throughput Data-Compression for Cloud Storage.* DATA MANAGEMENT IN GRID AND PEER-TO-PEER SYSTMES, 2010.

[3] Ivanchenko, Coughlan and Shen. *Real-Time Walk Light Detection with a Mobile Phone.* ICCHP'10 Proceedings of the 12th international conference on Computers helping people with special needs, 2010.

[4] Thompson, White, Dougherty and Douglas C. Schmidt. *Optimizing Mobile Application Performance with Model-Driven Engineering.* SOFTWARE TECHNOLOGIES FOR EMBEDDED AND UBIQUITOUS SYSTEMS, 2009.

[5] Google App Engine documentation - http://code.google.com/appengine/docs/

[6] Android Documentation - http://developer.android.com/reference/packages.html