Better Approach to Geofence Detection

Andrea Sghedoni MATR.0000736038*,
* DISI, University of Bologna, Italy

Email: andrea.sghedoni4@studio.unibo.it

***Abstract*—My Location Alert is a smartphone application developed for the Android operating system. The app allows you to create fences, selecting a center and a range, and capture all ENTER/EXIT events.**
Whenever app detect any of these events, it is automatically sent an alert SMS to a mobile (previously selected).
The main goal of the project, however, is to compare the battery consumption of two strategies/services for location monitoring in time.
The first strategy is based on the simple polling strategy, while the second adopt a smarter auto-adaptive approach to avoid an excessive battery consumption, which would make the app unusable.

I. Introduction

The geofencing approach means taking monitored geographical area and giving awareness to the device of the EXIT/ENTER/REMAIN events in one area. *Location-Based Services(LBSs)* are one of the most important cause of battery discharge in smartphones. If these services must to be continuous in time (with active jobs in background), it is probably the user's device pays a major deterioration of the battery. In recent years, Google is committed to the improvement of these activities, especially to reach an optimum point between precision and consumption. Please note that the location can be basically made from two mechanisms: the first concerns the use of the GPS sensor and has an important consumption of battery in the phone, while the second concerns the use of the network(or WiFi), even if in this precision is less, in comparison to the use of GPS sensor. Most of geofence mechanisms adopt a polling strategy, which controls the position at constant intervals (5 sec, 10 sec,...). This method, if it is merged with the use of GPS, drains battery in few hours. Considering also that the user, of course, does not use the smartphone only for geofencing, but developers must also take account of all the other apps that users install, use and need.
The main goal project is to implement an intelligent mechanism where current location is monitored frequently and with the GPS only if you are close to a fence, while controlling the position less frequently (and via network, to a lower consumption) as that the device moves away from the user-selected geofences. The approach is based on an auto-adaptive mechanism, where the algorithm tries to use certain resources only when necessary, details will still in Implementation Section. The app was developed as project part of Mobile Systems exam @ Alma mater Studiorum, Bologna.
The main use case of My Location Alert app is automatically alert a phone number(via SMS) of enter/exit events from a specific geographic area.

II. Related works

In GooglePlay Store there are several apps that deal with the theme of geofencing, one of them is *"EgiGeoZone Geofence"* where you can create geofences and choose between different actions(tracking, logging, email) to be made when enter/exit events are detected. Here you need to provide the center of the fence in terms of latitude and longitude, this approach is not completely user-friendly. Nowadays LBSs are becoming an important part of the Apps that we install every day on our smartphones, so developers need to be careful how they use these services and the effects they may have. This article, see [X] in references, provided the basis for learning to better the world of localization, especially show the differences between geolocation sources, such as networks, WiFi and GPS. Another interesting article is [Y], which shows proactive idea for the use of the different location technologies.
Can be seen as an important starting point from a developer can start to implement his own tracking strategy(or any other background service), taking account of having to manage different technologies at the same time.

III. Architecture

Basically, the app is composed with the two traditional components, such as frontend and backend. The first component is represented by the graphic interface where users interact with maps, registration forms and so on. The most interesting part is the back-end, *Figure I*, which is composed of: *Entity, SQLiteManager, Controller, Services*.

Entity map a fence in a java object that contains all the information needed to manage it, such as to address, latitude, longitude, range, state, activation, event to detect.
SQLiteManager manages the database, provides methods for CRUD(create, read, update, delete) data of the fence. This allows user to persist own preferences. The DB is composed of a single table, in which each entry corresponds to a fence, registered by users and each column map an attribute of the Java class Fence.
The *Controller* is an object that implements Singleton pattern, only one instance of that class is created and accessed. It exposes data and methods that should be available and accessed from anywhere in the project. Mainly, it provides the current user fence list and acts as interface to the SQLite DB, providing functions that implement CRUD verbs.

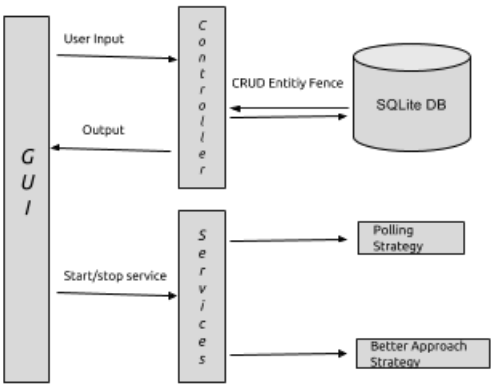

Fig. 1. Architecture

Services implement the monitoring and detection strategies of registred fences, which run in the background even when the app is not open. Only one of the two services can be active at a given time. The performance differences between the two services implemented is the true heart of the project, you will see in the Performance evaluation section that the polling startegy produces a battery consumption exaggerated relative to the other auto-adaptive strategy.

IV. Implementation

Android Studio is the official IDE for Android, where you can test code on different device emulators and monitor log. The development of a native Android App requires the use of object-oriented language JAVA. With regard to the persistence of the user information(basically the Fence objects with their attributes) it has been necessary to create and maintain a SQLite DataBase. We have used various services of GoogleMap Api Android, from the representation of the map(markers, fence circles) to use of location services(LocationUpdateRequests). In addition, the Geocoding service allows the user to select the center of fence, with an address and not with latitude and longitude. Git is used as a version control system, very important for step-by-step development and progress monitoring.

Location monitoring and fence events detection are the most important job that project must implement.

The first strategy acts polling, which get GPS position every 5 seconds(costant updates). LocationRequest, in this method, use $PRIORITY\_HIGH\_ACCURACY$ like priority parameter.

The second strategy is an auto-adaptive approach, which goal is a better low battery consumption. This method uses the most expensive resources, in terms of energy(such as a GPS sensor), only when absolutely necessary. This means that if the device is very close to the fences will access the location more frequently, looking for accurate precision. Conversely if user smartphone is away from fences, the position is monitored less frequently and not with the use of GPS sensor, but taking it from the network.

The approach is auto-adaptive and dynamically changes the frequency and precision of the location updates. The algorithm is based on 3 balaced parameters: *distance*(between fence and current location), *device speed* and *direction*. Each of these parameters are mapped on a scale from 0 to 5(see APPENDIX TABLES VI,VII,VIII), and multiplied by a coefficient, the total of the three parameters approximate sum indicates the frequency and the accuracy of the position updates, see TABLE I. The coefficients provide the importance that is given to each parameter(see 1a,b,c). The most influential parameter is the distance from the fence, but also the direction and speed parameters can become important in total evaluation, see (2).

$$\alpha = 0.8 \tag{1a}$$

$$\beta = 0.1 \tag{1b}$$

$$\gamma = 0.1 \tag{1c}$$

$$\boxed{f_i = \alpha f(distance) + \beta f(direction) + \gamma f(speed)} \tag{2}$$

f_i	Interval (second)	LocationRequest Precision
5	5	$PRIORITY\_HIGH\_ACCURACY$
4	30	$PRIORITY\_BALANCED\_POWER\_ACCURACY$
3	60	$PRIORITY\_BALANCED\_POWER\_ACCURACY$
2	180	$PRIORITY\_LOW\_POWER$
1	300	$PRIORITY\_LOW\_POWER$
0	480	$PRIORITY\_LOW\_POWER$

TABLE I
EVALUATION STRATEGIES

V. Performance evaluation

MyAlertLocation is tested on SO Android 6.0 and hardware device is LG Sprint. The removable Li-Ion battery has a nominal power of 2100 mAh, when fully charged (100%).

The device has been used only for app testing, no other app has been used in test period. This way is simple get precise estimates on battery power consumption and GPS utilization. Android 6, furthermore, allows to verify the consumption of these two factors for each application installed. The tests measure the difference between the polling strategy and the auto-adaptive method, in terms of consumed battery (mAh and percentage) and use time of the GPS sensor.

For each test is considered a time of 20 hours and a fence placed at a determined distance, respectively 300km, 10km, 1km.

In the end,app is tested when the user moves towards,in or out fence.

Table II shows the differences between the two strategies, considering a fence located approximately 300 km from the location of the device. If you use the Better Approach the energy consumption is only 0.04%, while the poll has an exaggerated consumption, close to 30%. The motivation is

that the service is looking for position every 5 seconds and constantly accesses to GPS, while the auto-adaptive method controls the position every 8 minutes, via the network with a very low battery consumption.
In Tables III and IV we can see that the polling strategy remains constant and produces an exaggerated consumption, while auto-adaptive strategy slowly increases its energy consumption according to the distance from the fence. This is obvious: if the distance from the fence is only 1 km, it controls the position more frequently(and with more accuracy) than 10 km. The use of GPS sensor increased because the distance to fence is only 1km and my observations should be more precise, so it might not be enough given the location via network.
The last test, Table V, is based on a scenario in which the user moves(entering and exiting from the fence within approximately 2km). Here, compared to the previous test (table IV), the consumption of battery(and GPS time) increases and this is an expected result, see Formula (2), because the user is moving in fence direction, with a certain speed so requires the position every 5 seconds, with the maximum possible accuracy(PRIORITY_HIGH_ACCURACY).
The battery saving, compared to polling, remains optimal.

Strategy	distance (km)	Time(h)	mAh usage	% battery	time GPS
Polling	300	20	800	38	20h
Better Approach	300	20	1	0.04	13s

TABLE II
TEST 1

Strategy	distance (km)	Time(h)	mAh usage	% battery	time GPS
Polling	10	20	800	38	20h
Better Approach	10	20	70	4	80s

TABLE III
TEST 2

Strategy	distance (km)	Time(h)	mAh usage	% battery	time GPS
Polling	1	20	800	38	20h
Better Approach	1	20	160	8	5min

TABLE IV
TEST 3

VI. CONCLUSIONS

This project shows that it is desirable to use the more expensive services(in terms of the battery) in an intelligent way, otherwise the user could drain battery in a few hours. In particular the use of the GPS sensor today is still an issue for smartphones, where some app abuse of GPS sensor.
The goal, in case of geofencing apps, is to use the services

Strategy	distance (km)	Time(h)	mAh usage	% battery	time GPS
Polling	IN/OUT-2km	20	800	38	20h
Better Approach	IN/OUT-2km	20	260	12	7min

TABLE V
TEST 4

in an intelligent way, using GPS only when necessary and if the device is located next to a fence. The improvements shown in the foo section is important, but there are margins of improvement. In the future a possible feature is to check the activities that the user is doing and regular location updates based on the results obtained(it is possible with Google ActivityRecognitionApi).
In the end, others improvements concern a commercial point of view, with a possible integration with leading social networks, such as Twitter, Facebook, Google+.

REFERENCES

[1] egmontr, **EgiGeoZone Geofence** *Android App on GooglePlay*, 2016.
[2] Axel Kpper, Ulrich Bareth, and Behrend Freese, **Geofencing and Background Tracking The Next Features in LBSs** *INFORMATIK*, Berlin, 2011.
[3] Axel Kpper, Ulrich Bareth, **Geofencing and Background Tracking The Next Features in LBSs** *IEEE 35th Annual Computer Software and Applications Conference*, Munich, 2011.
[4] **Creating and Monitoring Geofences** *Google Android Documentation*, 2016.
[5] **Receiving Location Updates** *Google Android Documentation*, 2016.

VII. Appendix

Distance(meters)	f(distance)
< 8000	5
8000-30000	4
30000-50000	3
50000-100000	2
100000-200000	1
> 200000	0

TABLE VI
DISTANCE EVALUATION

Speed(Km/h)	f(speed)
> 130	5
130-100	4
100-60	3
60-20	2
< 20	1

TABLE VII
SPEED EVALUATION

Direction	f(direction)
Yes	5
No	0

TABLE VIII
DIRECTION EVALUATION