International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:11, No:3, 2017

An Android Geofencing App for Autonomous Remote Switch Control

Jamie Wong, Daisy Sang, Chang-Shyh Peng

***Abstract*—**Geofence is a virtual fence defined by a preset physical radius around a target location. Geofencing App provides location-based services which define the actionable operations upon the crossing of a geofence. Geofencing requires continual location tracking, which can consume noticeable amount of battery power. Additionally, location updates need to be frequent and accurate or order so that actions can be triggered within an expected time window after the mobile user navigate through the geofence. In this paper, we build an Android mobile geofencing Application to remotely and autonomously control a power switch.

***Keywords*—**Location based service, geofence, autonomous, remote switch.

I. Introduction

SMART usage is booming in recent years. It is projected to have more than two billion smartphone users by the end of 2016 [1]. With the growing computation power and increasing communication throughput in both the devices as well as the internetworking infrastructure, the Location Based Services (LBS) applications are now possible for the general public. LBS applications provide information and/or services according to users' physical location [2]. LBS benefits from the advancements in mobile Internet access, positioning technologies, and rich user interfaces [3]. The first mobile device capable of LBS was Palm VII in 1999. The Weather.com app from The Weather Channel and the Traffic Touch app from Sony-Etak/Metro Traffic were made available based upon ZIP code positioning [4]. In June 2001 NTT DoCoMo revealed the first commercial global LBS service i-area for its i-mode handsets [5]. Today, there are trillions of daily LBS uses from control systems to smart weapons and could be one of the most heavily used application layer framework [2].

There are two types of LBS services in smartphone apps: pulled service and pushed service. The pulled services proactively pull information by sending requests for feedbacks based upon the device/user's location. With the pushed service, the location-relevant information is pushed or delivered to the subscriber autonomously as soon as the subscriber enters the geofence that relates to the messages' content. A geofence is a region defined by the radius around a target location or destination, e.g. 25 meters around a supermarket. Pulled service typically requires only one location update per event. Pushed service needs continual location updates for each event. In the context of LBS, pushed service is often referred to as geofencing where an action is triggered when the subscriber crosses the virtual fence [6].

Early geofencing development includes the where.com (now part of eBay) mobile app that delivers ads or coupons to its subscribers upon entering the perimeter of any of its 120000+ networking stores [7]-[9]. It can now track when a mobile subscriber enters one of its stores, or a competitor's store. The included scanner service can further suggest price comparisons between the store or eBay, and offer same-day delivery, ads, and/or coupons [10]. A more recent Geofencing Application is the geofencer [11]. Geofencer uses the Fused Location Provider API [12] to define a geofence, and choose when to be notified, how to be notified, and what actions to take.

There are three types of positioning techniques for location updates in mobile development: Global Positioning System (GPS), cellular networking positioning, and Wi-Fi positioning. GPS is the most common positioning technique with 32 satellites that continuously broadcast signals to the Earth surface. Smart devices use the embedded GPS receiver to receive signals from three to four satellites and calculate the current position through trilateration. Assisted-GPS (A-GPS) and its corresponding database servers can be deployed to achieve more efficient and accurate position fixes [13]. Cellular networking positioning uses the unique identifier of base stations which consists of the Mobile Country Code, Mobile Network Code, cell tower ID, and the corresponding Location Area Identifier. Each cell tower's unique ID (Cell ID) is coupled with a GPS fix. Smartphones use the built-in radio transceiver to receive the Cell ID to calculate the position. Wi-Fi positioning works in similar manner, but by using the location of Wi-Fi Access Point (AP) for a position fix.

Positioning techniques are measured on accuracy, availability limitation, energy consumption, precision, and time to first fix (TTFF). Accuracy is measured by deviation in meters from the actual location. Availability limitation denotes situations where position may not be fixed. Energy consumption can be measured in Watts seconds (Ws) or milli Watts seconds (mWs) per fix. Precision is the percentage of successful position fixes with the accuracy. TTFF is the time, in seconds, for the first position fix. Experiment [14] of A-GPS, Cell-ID, and Wi-Fi P shows the accuracy is 10m, 5km, and 50m, respectively. Limitation is indoors/canyons, regions with no cell coverage, and rural areas, respectively. Precision is 95%, 65, and 90%, respectively. Energy consumption is 6.616Ws, 1.013Ws, and 2.852Ws, respectively. TTFF is 15s, 3s, and 3s, respectively. While GPS positioning offers the best

Jamie Wong and Daisy Sang are with the Computer Science Department, California State Pollytechnic University, Pomona, CA 91768 USA.
Chang-ShyhPeng is with Computer Science Department, California Lutheran University, Thousand Oaks, CA 91360 USA (phone: 805-493-3819, e-mail: peng@callutheran.edu).

accuracy, it is the most power consuming technique. Cell-Id has the worst accuracy, but has the least power consumption. Wi-Fi positioning falls in between in both power consumption and accuracy. Mobile Android devices utilize both GPS and network for location updates, and network provider uses a combination of Wi-Fi AP or Cell-ID, with the preference to Wi-Fi AP [15].

This paper presents an Geofencing App (referred to as the Geofencing App) that autonomously and remotely toggles a switch on or off based upon the physical distance between the user/smartphone and the switch. Section II describes the app's architecture. Section III presents user interface. Section IV elaborates the UML and the control flow. Section V discusses the test cases and evaluations. Section VI concludes the paper.

II. Architect

The architect of this application, depicted in Fig. 1, includes several key components: The Geofencing App, a Wi-Fi router, If-This-Than-That (IFTTT) web service [16], and a Belkin WeMo switch [17].

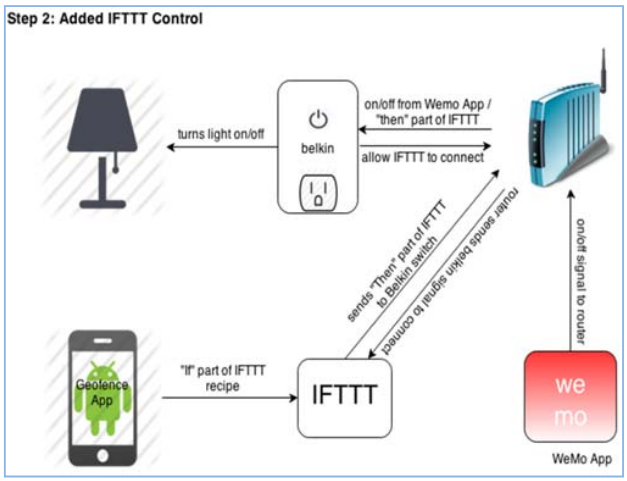

Fig. 1 Architecture of the Geofencing App

The Belkin WeMo switch is a smart device. Via a local router, the WeMo app enables users to toggle the switch on or off. If the router has Internet connection, then the WeMo switch can be controlled by the WeMo app remotely. The WeMo App has the option to connect to the If-This-Than-That (IFTTT) web service. IFTTT allows users to create conditional statements, called *recipes* [18], to trigger specified actions. Recipes have the format of “If event A happens, then event B will take place”. For example, the recipe to turn the WeMo switch on by sending an SMS text to IFTTT can be "If a specific SMS text message is sent to a specific phone number, then the WeMo switch will turn on". The connected device will go on accordingly.

Our Geofencing App is designed to autonomously turn on the switch when user enters the geofence and turn off the switch when user exits the geofence. This is achieved by saving the IFTTT recipes in an embedded SQLite database [19] and by executing the app in the background mode.

SQLite is a database that is embedded in Android smartphones. SQLite database has methods to create, delete, execute SQL commands, and perform other common database management tasks. SQLite is open sourced and requires limited memory. The background execution is implemented with the *started* and the *bound* services where a service is an application component that can sustain the operations without any user intervention [20]. The *started* service can be invoked by calling *startService()*. Once initiated, the service can run in the background indefinitely even if the caller is ended. Typically, the *started* service performs a single operation and does not return any results to the caller. The *bound* service, invoked by calling *bindService()*, provides a client-server mechanism that allows the corresponding application component to interact with the service via interprocess communication. A *bound* service runs only as long as another application component is bound to it. Multiple components can bind to the same service at any point, but the service will be terminated when all binding components unbind.

There are various techniques to conserve energy in mobile services, and they all fall within three optimization frameworks: GPS Substitution, GPS Reduction, and Hybrid Systems [21]. Instead of the GPS signals, GPS Substitution uses radio beacons from Wi-Fi AP, GSM, or UMTS Cell-IDs, or Bluetooth for location determination. GPS Substitution is most suitable for indoors applications where GPS signals is typically weak. GPS Reduction temporarily replaces GPS service with other positioning methods. One example is to use GPS only when the accelerometer detects sufficient movement [22]. The Hybrid System reduces energy consumption by using the most energy efficient method as long as accuracy is satisfactory [14]. Our Geofencing App adopts the practice of the hybrid system.

III. User Interface

There are several steps to prepare the entire application. First, the WeMo switch is registered with the IFTTT via a temporary activation pin generated by the WeMo app. Secondly the smartphone, which hosts the Geofencing App, is also registered with IFTTT via its online interface. Thirdly, from the IFTTT site, two recipes are created. Recipe 1, the Entering-Geofence recipe, has "If: SMS text #ON from phone, then: WeMo switch will turn ON". Recipe 2, the Exiting-Geofence recipe, is "If: SMS text #OFF from phone, then: WeMo switch will turn OFF". Afterwards, the Geofencing App takes full automatic control. The switch will be turned on or off when the user crosses the geofence boundary.

The Geofencing App can track multiple destinations. Upon the launch of the app, user is presented the main screen (a.k.a. main activity) as shown in Fig. 2.

The "Add Destination" button brings up the Add Destination activity screen, shown in Fig. 3. User can add a new destination by entering the address, by entering the latitude and longitude, or by pushing the "Save Current Destination" button.

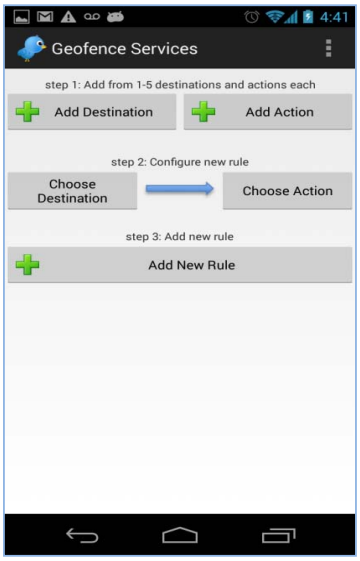

Fig. 2 Main Screen

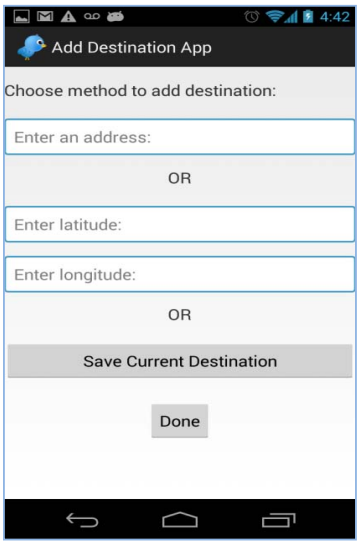

Fig. 3 Add Destination Screen

The "Save Current Destination" is a *bound* service, which enables multiple bindings from multiple activities and processes. It attempts to first use GPS provider for the initial position fix. If failed, it then tries network provider. If still unsuccessful, then the app will prompt a message to ask the user to physically move to a different location. Once the GPS location can be determined, the Geofencing App prompts the user with the detailed position data. Fig. 4 is an example.

When the new destination is determined by any one of the three choices, the push of "Done" button will bring up a dialogue (as shown in Fig. 5) for the user to name the location.

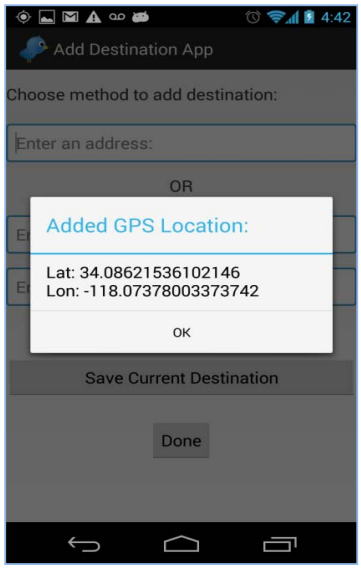

Fig. 4 GPS Location Confirmation

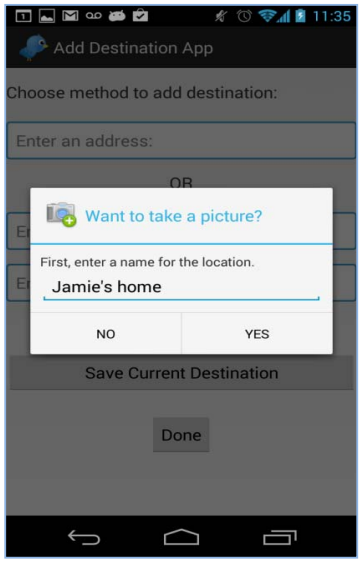

Fig. 5 Location Naming Dialogue

Next to the "Add Destination" button on the main screen is the "Add Action" button. The push of the "Add Action" button will display the Add Action screen (shown in Fig. 6) for IFTTT recipe. There are four *edittext* boxes. The first box asks for the name of the action. The second box is for the Entering-Geofence message. The third box is for the Exiting-Geofence message, and the last box is to enter the phone number which is to manage the messages. Two of the recipes for our testing are "If SMS text #ON to telephone number 14155740998, then turn the WeMo switch ON" and "If SMS text #OFF to telephone number 14155740998, then turn the WeMo switch OFF".

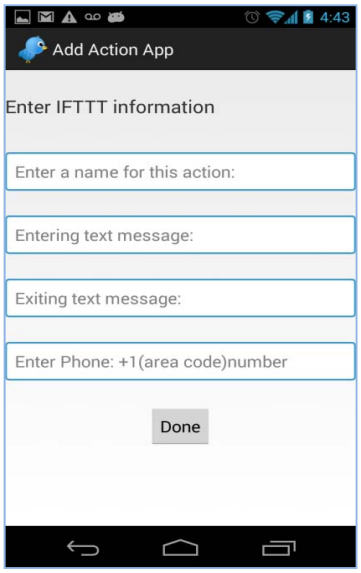

Fig. 6 Add Action Screen

Right beneath the "Add Destination" and "Add Action" buttons on the main screen are the "Choose Destination" button and "Choose action" button. As shown in Fig. 7, the "Choose Destination" button can bring up saved destinations in a list view. The "Choose Action" button can show saved actions, as depicted in Fig. 8.

Upon the selection of a destination and an action, the "Add New Rule" button on the main screen can then save the new rule as well as display all saved rules. Fig. 9 is an exemplary screen shot.

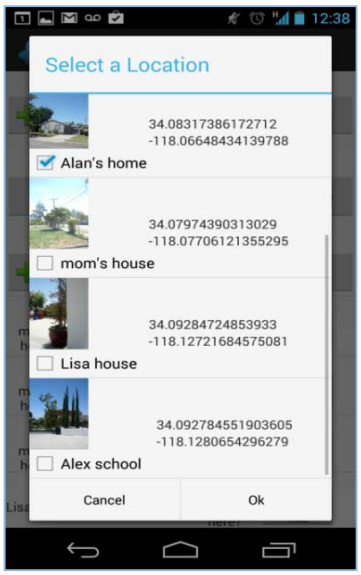

Fig. 7 Saved Destinations Screen

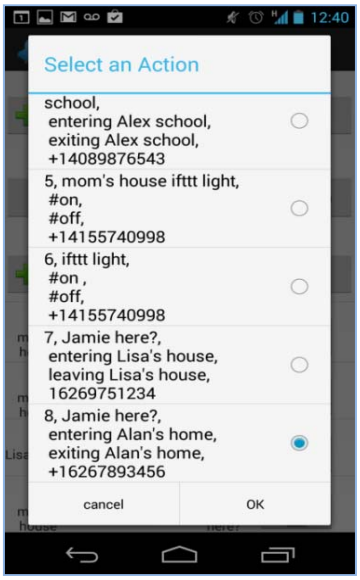

Fig. 8 Saved Action Screen

Fig. 9 Saved Rules Screen

There is an ON/OFF toggle button to the right of each rule. When a rule is turned ON, the corresponding service starts tracking this destination's geofence and activates the corresponding IFTTT recipes. Upon Entering-Geofence, an alert is displayed (shown in Fig. 10) and an SMS message is sent to IFTTT. When a rule is turned off, geofence tracking for the corresponding destination is stopped.

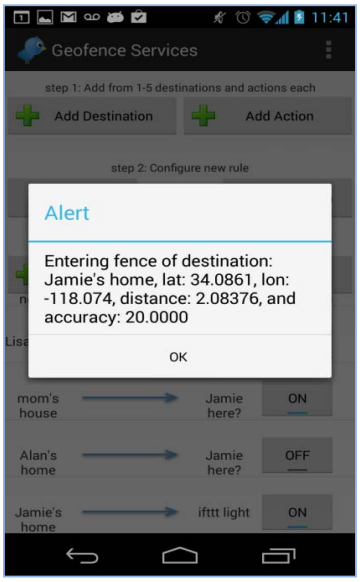

Fig. 10 Entering-Geofence Alert

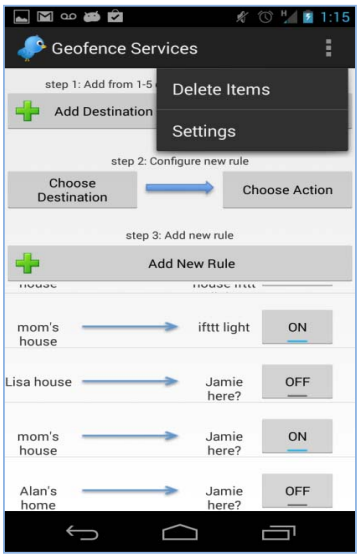

Fig. 11 Overflow Menu

As shown in Fig. 11, main screen's top right corner has *overflow* button, which comes with the choices of Delete Items or Settings.

Settings can be selected to change the basic user interface preferences. And, as shown in Fig. 12, Delete Items can delete Destinations, Actions, or Rules. Fig. 13 is a sample screen when user tries to delete one or more rules.

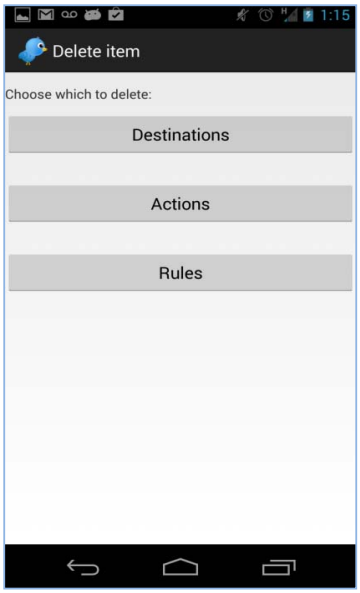

Fig. 12 Delete Items Screen

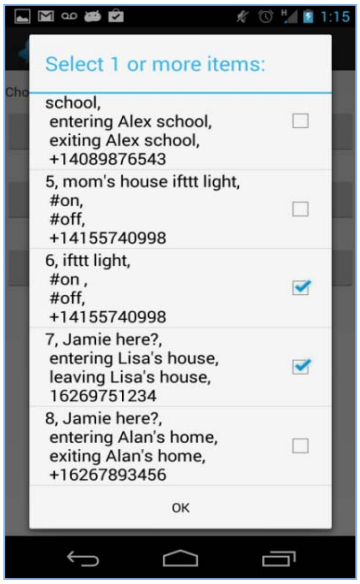

Fig. 13 Sample Rules Deletion Screen

IV. UML and Control Flow

Fig. 14 presents the UML of the Geofencing App. "GeofenceActivity" is the main activity when the app is launched. The "AddDest" class corresponds to the "Add Destination" on the main screen, which calls on the service class for a current location and the database class to store location data. The "AddAction" class corresponds to the "Add Action" on the main screen, which invokes the database class to store IFTTT recipes. The "DeleteItem" class corresponds to the "Delete Items" in the overflow menu. It calls on the database class to delete selected item(s). The "GeofenceBoundService" class communicates back to the "GeofenceActivity" via message handler.

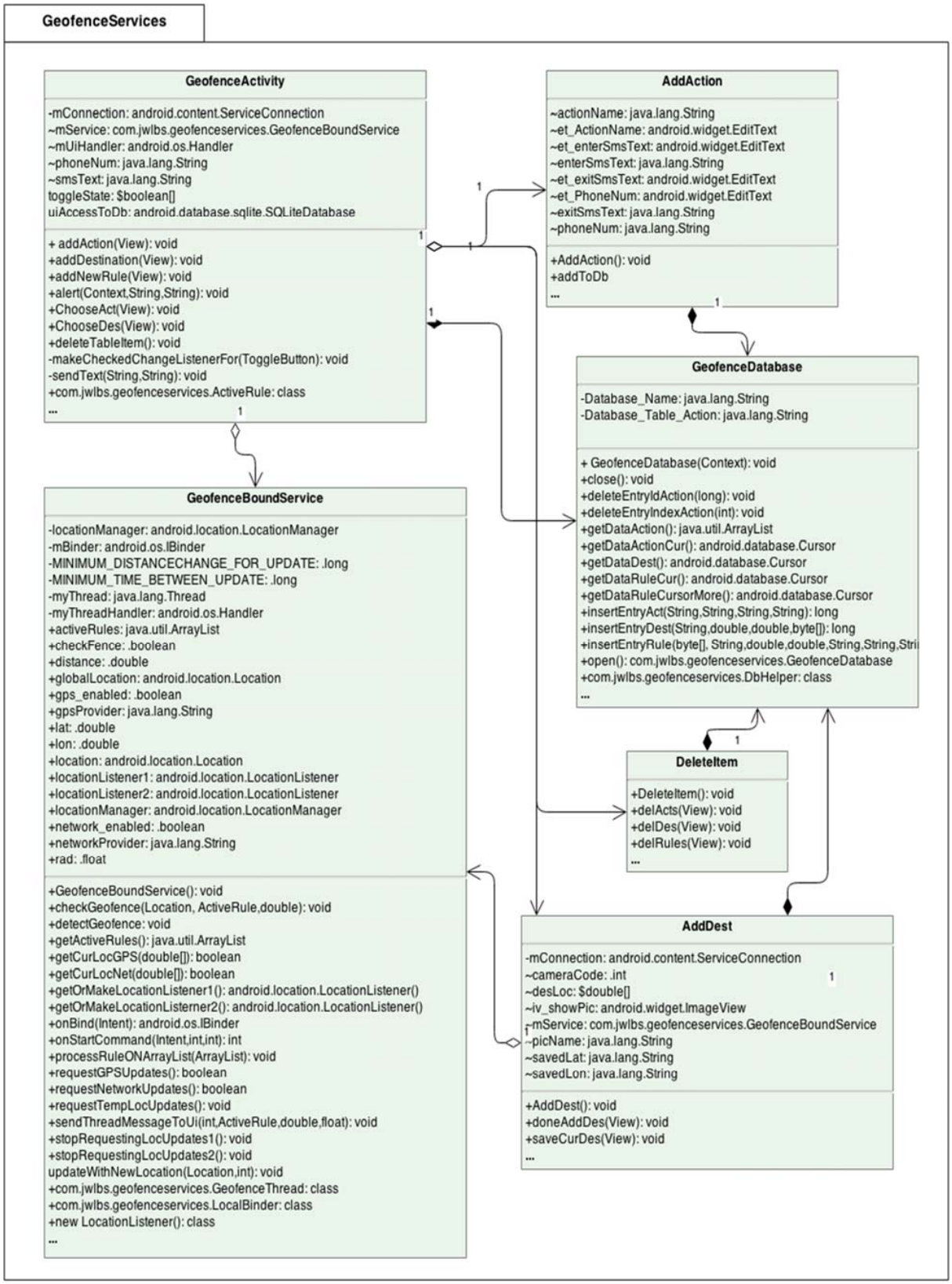

Fig. 14 The Geofencing App's UML

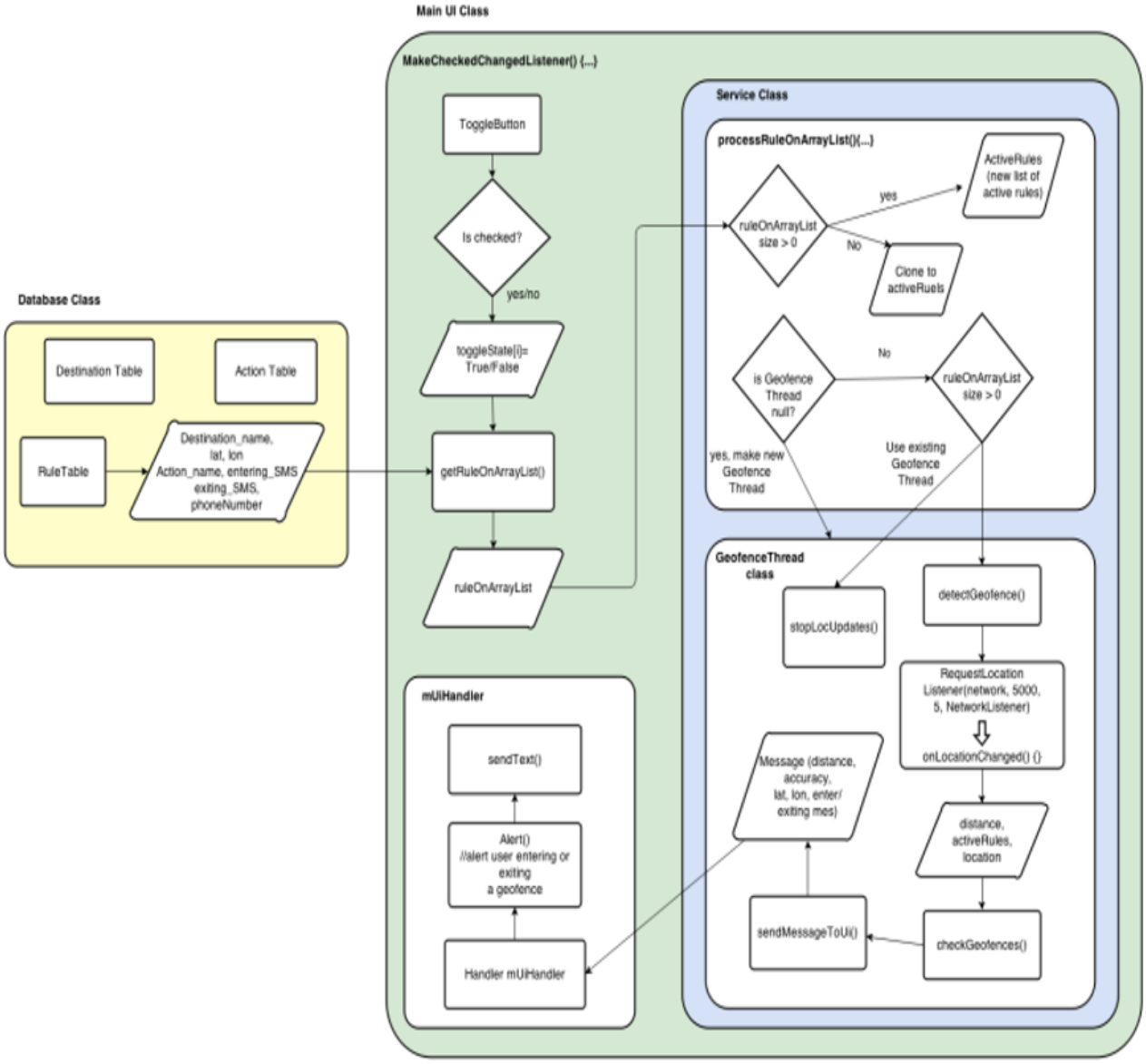

Fig. 15 Control Flow of the Main UI and Database Classes

Fig. 15 depicts the control flows in and between the main service class and database class. Upon the selection of a rule on the Save Rules Screen, details are fetched from the database.

Tracking of the relevant geofence is then initiated. Once the boundary of geofence is crossed, corresponding message will be sent and IFTTT recipe goes into action.

V. Testing and Evaluations

In the testing and evaluations, *network algorithm* denotes the use of network provider for location fixes, *GPS algorithm* denotes the use of GPS provider for location fixes, *correctness* refers to whether the switch is toggled upon crossing the geofence, and *accuracy* measures the actual distance between the destination and the current location when geofence is considered being crossed.

TABLE I
Case 1 Correctness and Accuracy

		Trial 1	Trial 2	Trial 3
Network	Entering-geofence	27m	20m	20m
	Exiting-geofence	61m	45m	55m
GPS	Entering-geofence	5m	5m	5m
	Exiting-geofence	5m	10m	30m

Three test cases, each with multiple trials, are conducted. For all cases and trials, Entering-geofence is set to 30-meter radius and Exiting-geofence is set to 60-meter radius. Case 1 compares the network algorithm and the GPS algorithm with one destination in a 12-hour window. Data from 3 trials are collected. Trial 1runs on network algorithm. Trial 2 runs on GPS algorithm. Both include 30 minutes on foot outdoors, 1 hour in driving, and the rest indoors. Trial 3 has similar route,

but with 2 hours on foot outdoors, about 90 minutes in driving, and the rest in doors. In all three trials of Case 1, switch is toggled correctly. Table I shows and compares the accuracies.

Case 2 also runs on both network algorithm and GPS algorithm with one destination, and measure the difference between indoors and outdoors during a 3-hour window. Wi-Fi signal is available only indoors. Correctness, accuracy, and power consumption are recorded. Table II shows the testing correctness and accuracy.

TABLE II
CASE 2 CORRECTNESS AND ACCURACY

		Correct?	Accuracy
Network Indoors	Entering-Geofence	Yes	20m
	Exiting-Geofence	Yes	25m
Network Outdoors	Entering-Geofence	Yes	20m
	Exiting-Geofence	Yes	23m
GPS Indoors	Entering-Geofence	Yes	5m
	Exiting-Geofence	Yes	15m
GPS Outdoors	Entering-Geofence	Yes	10m
	Exiting-Geofence	Yes	5m

Case 3 is designed to evaluate the app on one destination vs. multiple destinations. It runs on the network algorithm over a 12-hour window with two trials. Trial 1 has five destinations and trial 2 has one destination. For all cases and trials, the smartphone is reset to factory default before the Geofencing App is loaded. Phone is recharged to 100% for each run. For both the network and GPS algorithms, the frequency of location update is set to minimal time of 5 seconds and minimal distance of 5 meters. Results are shown in Table III.

TABLE III
CASE 3 CORRECTNESS AND ACCURACY

Destination		Correct?	Accuracy
1. Jamie's apt	Entering-geofence	Yes	45m
	Exiting-geofence	No	35m
2. Mom's house	Entering-geofence	Yes	45m
	Exiting-geofence	Yes	925m
3. Rosemead Library	Entering-geofence	Yes	31m
	Exiting-geofence	Yes	22m
4. Alex's school	Entering-Geofence	Yes	30m
	Exiting-Geofence	Yes	42m
5. Alhambra Library	Entering-geofence	Yes	33m
	Exiting-geofence	Yes	40m

In general, GPS clearly has much better accuracy. All IFTTT actions, except for Case 3 Destination 1, are triggered within 6 seconds after the geofence crossing is alerted. The possible cause of error for Case 3 Destination 1 is the high cement walls at the destination. For positioning with Wi-Fi signal, all results are within setting except for Case 3 Destination 3. This is the case when the Wi-Fi signal was lost and Cell-ID positioning steps in.

The other critical performance bench mark is the power consumption. Table IV shows the remaining battery in Case 1. It shows that the GPS algorithm's power consumption nearly triples that of the network algorithm.

TABLE IV
CASE 1 POWER CONSUMPTION

	% remaining battery		
	Trial 1	Trial 2	Trial 3
Network	76%	77%	78%
GPS	32%	32%	25.50%

Table V shows the results of Case 2. The network algorithms indoors trials are 1.8 times more efficient than outdoors. GPS algorithm indoors trials are 1.4 times more efficient than outdoors. For the indoors trials, network algorithm is 3.4 times better than GPS. For the outdoors trials, network algorithm still outperforms GPS by 2.67 times.

TABLE V
CASE 2 POWER CONSUMPTION

	% remaining battery	average power consumption
Network Indoors	95% (consumed 5%)	1.67 (%/hour)
Network Outdoors	91% (consumed 9%)	3 (%/hour)
GPS Indoors	83% (consumed 17%)	5.67 (%/hour)
GPS Outdoors	76% (consumed 24%)	8 (%/hour)

Table VI compares the two trials in Case 3. Case 3 shows interesting results. Trial 1 has five times more destinations but consumes only 6% more power. Both trials fix the positions at the same frequencies, but trial 1 repeats similar tasks over five different destinations. This coincides with the findings in [23], which highlights the side effect of operation/activity repetition in smart devices' power consumption.

TABLE VI
CASE 3 POWER CONSUMPTION

% remaining battery	
Trial 1 (5 destinations)	Trial 2 (1 destination)
69%	75%

VI. CONCLUSION

The Geofencing App enables continual location racking for multiple destinations through a simple and user-friendly interface. Source code is available online [24]. The Geofencing App is superior than the commercial app such as WeMo app in that user intervention is unnecessary after initial setup. In general, geofencing activities need to achieve a fine balance between energy efficiency and positioning accuracy in order to be useful and meaningful to general consumers. There are some significant ongoing improvements in the underlining platform and operating system. Coupled with the ever growing IFTTT services, further advanced geofencing applications can be expected.

REFERENCES

[1] Number of Smartphone Users Worldwide from 2014 to 2019, Statista, http://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/, last access 2016.
[2] A. Dey, J. Hightower, E. Lara, and N. Davies (2010). "Location-Based Services", *IEEE Pervasive Computing*, vol. 9, pp. 11-12, Jan. 2010.
[3] Location Based Services, Nauipedia, http://www.navipedia.net/index.php/Location_Based_Services, last access 2016.

[4] Location-Based Service(LBS), MediaBUZZ, http://www.mediabuzz.com.sg/asian-emarketing/may-2011/1264-location-based-service-lbs, last access 2016.
[5] NTT DoCoMo LBS service *i-area*, http://www.itu.int/itunews/issue/2001/08/licensing3g.html, last access 2016.
[6] A. Greenwald, G. Hampel, C. Phadke, and V. Poosala, "An Economically Viable solution to Geofencing for Mass-Market Applications", Bell Labs Tech. J., vol. 16, no. 2, pp. 21–38, 2011.
[7] PayPal Media Network, https://advertising.paypal.com/#mobile-targeting, last access 2016.
[8] E. Schonfeld, Where Is Awarded the Mother of All Geofencing Patents, TechCrunch, http://techcrunch.com/2010/12/21/where-Geofencing-patent/, last access 2016.
[9] R. Kim, Local, Mobile Ambitions Driving eBay's Purchase of Where, Gigaom, https://gigaom.com/2011/04/20/local-mobile-ambitions-driving-ebays-purchase-of-where/, last access 2016.
[10] eBay Acquires Location-Based Media and Advertising Company Where, http://techcrunch.com/2011/04/20/ebay-acquires-location-based-media-and-advertising-company-where/,last access 2016.
[11] Geofencer, https://play.google.com/store/apps/details?id=com.arpacell.fencer,last access 2016.
[12] Location Aware APIs, http://developer.android.com/google/play-services/location.html, last access 2016.
[13] Assisted GPS, GPS World, http://gpsworld.com/innovation-assisted-gps-a-low-infrastructure-approach/, last access 2016.
[14] U. Bareth and A. Kupper, "Energy-Efficient Position Tracking in Proactive Location-Based Services for Smartphone Environments", in *Proc. IEEE 35th Annual Computer Software and Applications Conference*, pp. 516–521, 2011.
[15] Network Provider in Android, Stackoverflow, http://stackoverflow.com/questions/14790269/network-provider-in-android, last access 2016.
[16] If-This-Than-That Web Service, IFTTT, http://ifttt.com, last access 2016.
[17] Belkin WeMo Switch, Belkin, http://www.belkin.com/us/F7C027-Belkin/p/P-F7C027;jsessionid=FCEC94F8E035E2CCD6B60033B63FFCF3/, last access 2016.
[18] Recipes, IFTTT, https://ifttt.com/recipes, last access 2016.
[19] SQLite Database, SQLite, http://www.sqlite.org/, last access 2016.
[20] Services, Android Developers, http://developer.android.com/guide/components/services.html, last access 2016.
[21] U. Bareth, "Simulating Power Consumption of Location Tracking Algorithms to Improve Energy-Efficiency of Smartphones", in *Proc. of IEEE 36th Annual Computer Software and Applications Conference*, pp. 613–622, 2012.
[22] T. O. Oshin, S. Poslad, and A. Ma, Improving the Energy-Efficiency of GPS Based Location Sensing Smartphone Applications, in *Proc. of The 11th International Conference in Trust, Security and Privacy in Computing and Communications*, pp. 1698–1705, 2012.
[23] S. Lafond and J. Lilius, "An Energy Consumption Model for Java Virtual Machine", TUCS Technical Reports 597, Turku Centre for Computer Science, 2004.
[24] The Geofencing App Source code, https://goo.gl/Cj3D9i, last access 2016.