BIKE TRACKER

By,
Shah, Prachi
Vedre, Jothin Reddy
Mandapati, Siddhardha Raju

CSCI - P535 Pervasive Computing
Final Report
2014

1. INTRODUCTION

Student/Faculty/Staff at Indiana University, Bloomington spend a lot of money in order to park their personal vehicles on-campus. As an alternative, they use bicycles. A lot of IUB students/faculty/staff use bikes due to convenience, need for less on-campus parking space and costs. But, what if a bike theft occurs? According to Indiana Daily Student newspaper, there have been several bike thefts within and off-campus in Bloomington [6]. Incidents like multiple Little 500 bikes stolen from Black Key Bulls and Teter teams of the Little 500, a $1,500 bike stolen from Third Street in front of Memorial Hall are prevalent [8]. These bikes are stolen despite the bike locks. Since, IUB is famous for the Little 500 bike race; a lot of students prefer using bikes that are sophisticated. Such bikes are costly and a frequent target for thefts. Bike theft is common not only at IUB, but at several universities across the United States. Bike theft is a common and important problem at IUB which needs to be addressed as people at IUB rely a lot on their bikes for daily travel, these bikes are sophisticated and costly and, these bikes are used by Little 500 teams. Indiana University Police Department receives several stolen bike reports every year and recommends several bike theft prevention measures [9]. Hence, we need a mechanism to notify the bike owner about a possible bike theft in real-time and not after the theft has occurred. This is required because, not only are these bikes vulnerable to thefts, but they are also a frequent target to thefts.

2. RELATED WORK

Spybike bicycle trackers:

Spybike bicycle trackers are covert GPS devices that are mounted on the users' bike, arm the bike when the user locks it and notifies the user of a possible bike movement [5]. The user can track the bike online or on his/her phone and find out the current location of the bike using a free real-time tracking service. This is a costly service that relies on GPS services to track the bike location. Instead a device which supports Wi-Fi can be used so as to reduce the high costs of using a mobile data service.

3. OUR APPROACH

To address this problem, we propose a notification system to track the bikes on campus and make the bike owners aware of a possible bike theft. We do not implement any mechanisms to prevent a potential bike theft. We propose using an Android based smart mobile phone to track the bike and send its location details to the bike owner.

The primary smart phone is mounted on the bike, the secondary smart phone is the personal mobile phone used by the bike owner. The primary smart phone can be replaced by a location sensor. The owner needs to activate bike tracking on the mounted smart phone. Two buttons are provided on the user-interface of the mobile phone mounted on the bike for the user to activate or deactivate the tracking service. On activation, an Android service is initiated. This service is used to trigger an activity every time there is a change in the status of the Wi-Fi connection. On disconnection from a Wi-Fi network, the service triggers an SMS manager which will notify the user about the change in location of the bike. On connection to a different Wi-Fi network, the service triggers an asynchronous task which captures the MAC addresses of the access points within range and obtains the location co-ordinates using this information [14]. Once the asynchronous task ends, an SMS with a link is sent to the user. On clicking the link, the owner will be redirected to Google Maps application that will plot a marker on the exact location of the bike.

With every new access point connection establishment and disconnection from the old access point, the service sends an SMS to the bike owner with the updated location details. There are two types of alerts. First, a simple alert informing the owner of an access point disconnection and the second is a detailed alert with the new location co-ordinates when a new connection is made.

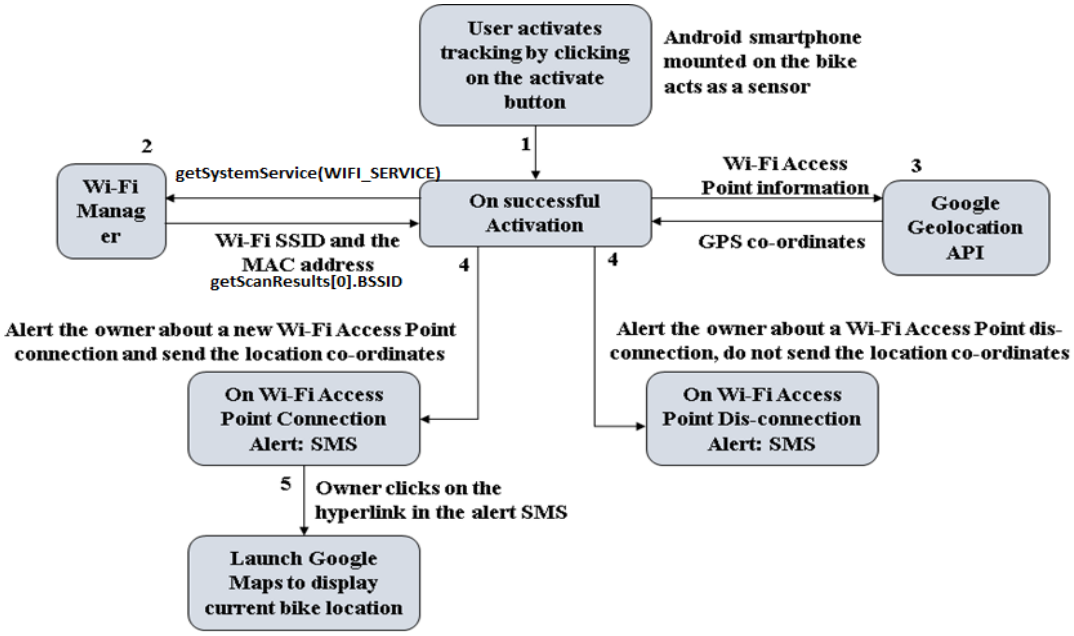

Figure 1. Process model

Our approach is a low cost, low maintenance, easily mountable sensor device that uses Wi-Fi services to track the location of the bike in real-time. Also, using Google Maps Geolocation API to detect the current location produces detailed and accurate location details making it a sophisticated bike tracking service.

4. USER INTERFACE

On launching the application, there are two buttons available to the user, “Start Tracking” and “Stop Tracking” which can be used to activate / de-activate the tracking service.

Figure 2. Buttons to start and stop the Bike Tracker service

Once the tracking service is activated, the service keeps listening for Wi-Fi connection/disconnection events and sends the appropriate SMS to the user.

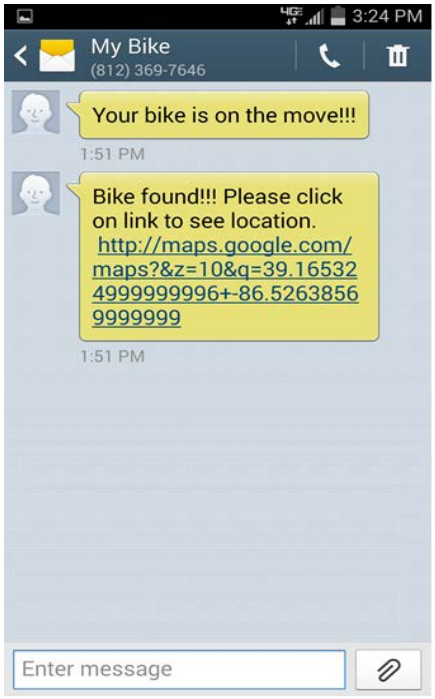

Figure 3. SMS Alert to the end user with current bike location details and a link to Google Maps

The SMS shows that the bike has been stolen if the trigger was Wi-Fi disconnection. If the trigger was Wi-Fi connection, the SMS shows that the bike has been found, along with a link which will display the location on Google Maps as follows:

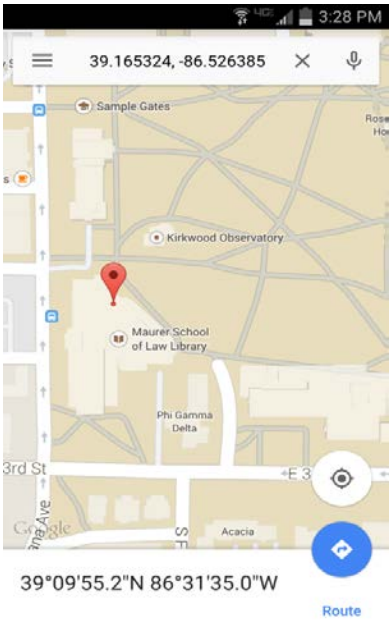

Figure 4. Google Maps displays current bike location to the end user

5. IMPLEMENTATION

We implement a proof of concept prototype using an Android smart phone. We develop this prototype using Eclipse IDE with Android development kit. We consider the change in Wi-Fi connectivity as is the sensed contextual data. It indicates a change in the location of the bike. In addition, the user enters his/her mobile number to receive notifications. Every new location is a new situation and will have relevant sensed data. The prototype uses the following components:

Service:

A service is a long-running operation that can be utilized by other applications. Services do not require user interaction. The service will capture all the Wi-Fi access point SSID's or MAC addresses that the phone detects and use them to construct a JSON object which will be sent in the request to the API. Google's Geolocation API uses JSON data format for both request and response, and the content type of both is application / JSON [15].

BroadcastReceiver: A BroadcastReceiver is an Android component which is used to register and listen to system events. An event in this case is the change in the status of Wi-Fi connectivity.

Within the BroadcastReceiver, we check the network information obtained from the ConnectivityManager object, to detect if the event is triggered due to a Wi-Fi connection or disconnection.

ConnectivityManager: A ConnectivityManager is an Android component that will be used to capture and manage various parameter of a Wi-Fi connection. This class will fetch the list of current active configured networks (SSID and so forth) to make decisions about which network to connect to, capture dynamic state information of the current active Wi-Fi connection and determines various Intent actions' broadcasts triggered as a result of any change in the state of the Wi-Fi connection.

```
ConnectivityManager connManager = (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
NetworkInfo networkInfo = connManager.getNetworkInfo(ConnectivityManager.TYPE_WIFI)
if (networkInfo.isConnected()) {
    // Invoke the service
}
```

If the event was triggered by Wi-Fi connection, we obtain the location details and then send these details in an SMS/E-mail. If the event was triggered by Wi-Fi disconnection, we send the last known location details in an SMS/E-mail.

Access points:

WifiManager: Our service will use the WifiManager class to identify the MAC addresses of all the access points within range. This class determines connection details of Wi-Fi networks within range.

getScanResults(): This is a public method of the WifiManager class that returns the list of all scanned Wifi networks within range. Each ScanResult object contains information about the access points, which can be used to obtain information such as the MAC address, signal strength etc.

We get the SSID of all the Wi-Fi networks within range and use them to retrieve the location details. The below code is used to obtain the details.

```
WifiManager w = (WifiManager) context.getSystemService(Context.WIFI_SERVICE);
List<ScanResult> scanResults = w.getScanResults();
result = scanResults.toString();
for(ScanResult s : scanResults) {
    JSONObject wifiObject = new JSONObject();
    wifiObject.put("macAddress", s.BSSID);
    wifiObject.put("signalStrength", s.level);
    wifiObject.put("channel", s.frequency);
    wifiList.add(wifiObject);
}
```

Location details:

Every time the device connects to a new access point, certain information is captured. The MAC addresses of the access points within range are noted and the location co-ordinates using the Geolocation API are obtained. The scope of the project is to locate the bikes within IUB's campus. The location data obtained from the API are latitude and longitude co-ordinates using which we form a link to plot the location on Google Maps. This appropriate location details are sent to the owner via an SMS.

The Google Maps Geolocation API: The API returns a location (latitude and longitude) and the accuracy of the co-ordinates, based on the information of the Wi-Fi access points that the mobile device detects. Communication between the mobile device and the API is done over HTTPS using POST method. The request and the response are formatted as JSON objects. We are using Google Geolocation API for Business' API key with the request.

We create a JSONObject for every Wi-Fi access point ScanResult and store the respective MAC address (BSSID), signal strength and frequency channel.

```
JSONObject wifiObject = new JSONObject();
wifiObject1.put("macAddress", scanResult.BSSID);
```

Next, we create an ArrayList of JSONObject type that stores all these access points' data.

```
ArrayList<JSONObject> wifiList = new ArrayList<JSONObject>();
wifiList.add(wifiObject);
```

An HttpClient is used to create an HttpPost object that contains the JSON with access point details. This object is then sent in the request to the Geolocation API using the following command.

```
client.execute(post);
```

Upon executing this command, we receive a response from the service, with the output JSON from Geolocation API. A sample JSON output from the Geolocation API will be similar to the following representation [15]:

```
{
    "location": {
        "lat": 39.170541,
        "lng": -86.494257
    },
    "accuracy": 11000
}
```


location: Estimated latitude ("lat") and longitude ("lng") of the bike, in degrees.
accuracy: Accuracy of the estimated location in meters. It is the radius of a circle around the estimated location.

SMS alerts:

SmsManager: This class will be used to send text messages to the bike owner's mobile phone. *sendTextMessage()* method will specify the mobile number and the text of the message being sent to the owner. The phone number of the owner will be statically defined in this method. The SMS received by the user contains the event type of whether it is a Wi-Fi connection or disconnection and also location details embedded into a link. Clicking on the link displays the bike location in the Google Maps application on the user's phone.

6. FEASIBILITY

Technology and system feasibility:

Platform: The proposed project is feasible as it uses a programmable Android smart mobile phone as a bike tracking device. There is no version dependency; however, we propose using API level 19 which is Android 4.4 (KITKAT) version. The phone used by the bike owner can be any phone that supports text messaging and/or an e-mail client. (Android, iOS, Windows phone and so forth).

Operational feasibility:

Reliability: The proposed solution is reliable as it is based on a smart, reliable, platform-independent and adaptable Android OS.

Time taken to update new location: A sum of the time taken with the mobile phone to connect to the nearest active Wi-Fi connection, fetching the MAC address, capturing the location details and sending a text message/ e-mail to the owner will account for the total time required to track the current location of the bike.

Economic feasibility:

Our solution is neither inexpensive as compared to some existing solutions nor as expensive as some of the relatively sophisticated solutions such as the Spybike tracking system [5]. However, our approach provides a proof of concept. Our solution uses Wi-Fi services which are cost effective compared to solutions that use high cost GPS services. The solution is economical, as only one Android smartphone mounted on the bike will be used for tracking. The phone wouldn't need support for GPS, since it only uses Campus Wi-Fi network. The phone used by the owner is not considered as a part of this feasibility study, since practically most people today have personal smart phones.

In addition to relatively low cost of the proof of concept prototype, we would greatly reduce the cost of the application if we use the alternative tracking mechanisms as mentioned in section 6.

7. ALTERNATIVE APPROACHES

As an alternative approach, we suggest using either of the two below mentioned sensing devices to make the application more feasible, economically.

7.1. Wi-Fi Detective: Wi-Fi Finder with LCD:

The Wi-Fi Detective instantly detects nearby wireless hotspots [10]. It also acts as a wireless network adapter that turns a wireless station into a WLAN access point. This is a rechargeable, USB 2.0 compliant device that offers an LCD screen to display details about wireless hotspots within the range. Compliant with IEEE 802.11 B/G 2.4GHz standards, Wi-Fi Detective detects several hotspot details like the SSID, network signal strength, network type (IEEE 802.11 B/G), network mode (ad-hoc or infrastructure), operating channel and, the number of remote access points detected [11]. The device is accompanied by a driver that can be installed to program the Wi-Fi detective.

7.2. Monnit Wireless Accelerometer:

It is a 900 MHz wireless accelerometer that can detect impact, vibration, inclination, and so forth [12]. Once activated in a user-defined time interval, the device measures the gravitational force along X, Y and Z axes and display the gravitational force data at each axis [13]. The device is mountable and rechargeable.

Although our approach is limited to an Android smart phone, using one of the above two sensors is comparatively economically feasible. Also, these devices are compact, portable, easy to mount and rechargeable. These features increase the duration of their use. Hence, we suggest using devices similar to the aforementioned sensing devices to track the bike in real-time.

8. EVALUATION

We deployed our application on an Android smartphone and activated the tracking service before mounting it on a bike. We then moved the bike to several locations on-campus. We noted every SMS which is sent to the user and evaluated the application based on the following measures:

8.1. Detection of change in Wi-Fi connectivity:

We have noticed that the application accurately detected the change in the status of Wi-Fi connectivity. We have made this observation following the results seen on N. Woodlawn Avenue between Informatics building and the Indiana Memorial Union. However, the user has been receiving multiple text messages with the same message. The BroadcastReceiver which we register to listen to Wi-Fi connectivity events, receives multiple intents, resulting in service being called multiple times. As a result, multiple text messages are sent to the user.

8.2. Accuracy of location updates:

We have noticed that the location updates sent out by the application were not up to the mark in all the results observed. When our bike was near Informatics or Wells Library, the application pointed the bike to be 0.3 miles away, at the Arts Museum. Similarly, when the bike was near Lindley Hall, the application pointed the bike to be 0.1 miles away, at the Maurer School of Law.

9. DISCUSSION

Our evaluation showed that it is possible to track a bike using the campus Wi-Fi service. However, the accuracy of Google's Geolocation API didn't turn out to be as good as expected.
Two issues were observed during our evaluation. They are multiple SMS being sent for single Wifi connection/disconnection event and location co-ordinates obtained from the geolocation API are inaccurate by several meters. We have come up with solutions to these two problems.

Multiple SMS: The BroadcastReceiver which we implemented, listens to two kinds of intent actions; STATE_CHANGE and WIFI_STATE_CHANGED, which are registered in the android manifest file. We observed that STATE_CHANGE event listens to events such as Wi-Fi connection and disconnection, while WIFI_STATE_CHANGED listens to events such as Wi-Fi connection, disconnection, enabled, disabled, connecting etc. Since the events for Wi-Fi connection/disconnection are registered by both the intent actions, multiple events are getting triggered, resulting in multiple SMS. We can solve this by registering only the STATE_CHANGE intent action for the BroadcastReceiver, thus listening to a single event for Wi-Fi connection and disconnection.

Location Accuracy: The accuracy issues from Google's geolocation API can be prevented by creating a custom database containing the list of MAC address and actual location of all Wi-Fi access points across the campus. To generate this list, we need to obtain the actual location co-ordinates of each Wi-Fi router by using a GPS device placed near it. This list can be maintained on a server. To track a bike, we can send the MAC addresses of all the Wi-Fi networks in range of the device and obtain the list of location co-ordinates for these MAC addresses. By aggregating the latitude and longitude values of the returned list, we can obtain the accurate location of the bike.

10. FUTURE IMPROVEMENTS

An advanced solution that stores information regarding recent thefts, location / time of occurrence of the incidents and sending weekly/bi-weekly reports to the Indiana University Police Department about the thefts. Also, a tracking system for University Bike rental services to track their bikes. Using Pattern-based motion detection we can follow a pattern of the bike's motion. For example, a bike in motion for 10 seconds indicates a significant change in the bike location. Coupled with a change of the access point, we can determine the bike has been moved in absence of the owner and can possibly imply a bike theft. If there is a change in the motion of the bike for less than 10 seconds with no change in the access point then we can conclude that the bike is at the same location it was parked at. This is a normal behavior. Any outliers like the one mentioned above can detect a possible bike theft. This motion-based pattern can be used to improve the accuracy of the tracking and handle false negatives.

11. GLOSSARY

Access point - Wireless device to connect to Indiana University Bloomington's wired network named *'IU Secure'* using Wi-Fi technology [1].
Android OS - As in Google developed *'Android (operating system)'* [2]
API - Application Programming Interface
IUB - Indiana University Bloomington
MAC address - Media Access Control address [3]
Map - As in *'Map'* data structure in Android
SMS - As in *'Short Message Service'*
SSID - As in *'Service Set Identifier'* [4]
Wi-Fi - Local area wireless technology

12. REFERENCES

[1] About IU Secure wireless. https://kb.iu.edu/d/awws
[2] Android. developer.android.com
[3] MAC address. en.wikipedia.org/wiki/MAC_address
[4] SSID (Service Set Identifier). www.techterms.com/definition/ssid
[5] SpyBike. http://www.integratedtrackers.com/
[6] Indiana Daily Student. http://www.idsnews.com/
[7] http://www.idsnews.com/blog/hoosierhype/2009/03/bikes-stolen-from-bkb-teter
[8] http://www.idsnews.com/article/2007/04/update-330-p-m-1500-bike-stolen-in-armed-robbery
[9] Bicycle Theft Prevention. http://www.indiana.edu/~iupd/crimeBikeTheft.html
[10] Wi-Fi Detective: Wi-Fi Finder with LCD. http://www.startech.com/Wi-Fi-Detective-Wi-Fi-Finder-with-LCD~WIFIDETG
[11] Wi-Fi Detective: Wi-Fi Finder with LCD.
http://www.startech.com/media/products/WIFIDETG/PDFs/WIFIDETG_Datasheet.pdf
[12] Accelerometer – Tilt. http://www.monnit.com/Products/Wireless-Sensors/Commercial-Coin-Cell/900-MHz/Wireless-Accelerometer-Tilt-Sensors
[13] Monnit Wireless Accelerometer.
http://resources.monnit.com/content/documents/datasheets/commercial/MD011-Accelerometer-Data-Sheet.pdf
[14] The Google Maps Geolocation API.
https://developers.google.com/maps/documentation/business/geolocation/
[15] The Google Geocoding API.
https://developers.google.com/maps/documentation/business/geolocation/