2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress

Energy Efficient Trajectory Recording of Mobile Devices Using WiFi Scanning

Maroš Čavojský
Faculty of Electrical Engineering and Information Technology, Slovak University of Technology, Bratislava, Slovakia
Email: maros.cavojsky@stuba.sk

Martin Drozda
Faculty of Electrical Engineering and Information Technology, Slovak University of Technology, Bratislava, Slovakia
Email: martin.drozda@stuba.sk

***Abstract*—Efficient user location tracking is a key prerequisite of many mobile applications that aim at A/B, usability or Quality of Experience tests. The purpose of these tests may require that location tracking is energy efficient and transparent to the user. Even though most mobile devices are equipped with a GPS receiver, its frequent use can cause a high battery drain. Our design of continuous location tracking aims at providing a reasonable trade-off between energy consumption and location acquisition accuracy. We use WiFi scanning for decisions about when to start using GPS receiver and as the user moves we adjust sensing approach for best fit to current situation scenario. Our experiments were performed with Android powered smart phones. The experiments show that our design provides significant energy savings when compared to other continuous location tracking alternatives. We also provide a detailed discussion of challenges and existing techniques for user location tracking.**

1. Introduction

Nowadays many smart phones are equipped with positioning devices such as Global Positioning System (GPS) for providing an accurate location. GPS requires line of sight to GPS satellites. For GPS, time to obtain the first device location is called Time To First Fix (TTFF). Since TTFF may take longer time and consume a lot of battery power, location acquisition is often complemented with Assisted Global Positioning System (A-GPS). A-GPS sends IDs of available nearby cell towers and WiFi hot spots over the Internet to a location server (such as the Google Location Server), a database with location information on WiFi hot spots and cell towers. Location server replies with approximate satellite locations what can significantly decrease TTFF. A-GPS requires an active Internet connection.

Continuous collecting of device locations can lead to high battery drain. When done with GPS, it may result in complete mobile device discharge within 4-6 hours. Due to this, we propose and analyze an approach for user trajectory tracking with reasonable energy-accuracy trade-off. Our approach can be described as cascade location detection, where the first member of the cascade chain is based on WiFi scanning. If a location change gets detected then GPS based location acquisition is attempted. Since WiFi scanning comes at a much lower cost than GPS, this approach has the potential to save battery power, especially in cities where we can expect a high density of WiFi access points. This gives rise to a trade-off between location precision and energy efficiency, since location acquisition with GPS can be significantly delayed.

Cascading classification in settings such as network intrusion detection, mobile operating system failure detection and error detection in ad hoc wireless networks has been extensively investigated; see [1] and references therein. Trade-offs that cascading classification offers have been investigated by Rokicki and Drozda [2]; their evaluation approach was based on iso-performance.

2. Related Work

Location providers are the sources of location data in Android. There are two main location providers: (i) the GPS location provider and (ii) the network location provider. Since they use different ways to obtain location data, they differ in accuracy, minimum notification interval, TTFF, and battery consumption. After analyzing location information and combining all location providers we can say, that there the following types of location data to which we have programmatic access [3]:

- *Cached GPS*: most of Android devices have the capability of caching the last retrieved GPS location. This is typically used when an application first starts up, in order to prevent from waiting for TTFF.
- *Cached Network*: similar to cached GPS, Android devices can also cache the last retrieved location determined by the cellular network location provider.
- *Cached Real-time Network*: this provider collects IDs from the cell networks and WiFi hot spots in the area, then searches in local device database to determine an approximate location. This is not available in all countries and works reliably only on places for which device retrieved Real-time Network in recent days. Depending on carrier, you can get

978-1-5090-2771-2/16 $31.00 © 2016 IEEE
DOI 10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.107

access to time-stamp, latitude, longitude, altitude, accuracy and speed.

- *Real-time GPS*: location information that is streamed from the GPS receiver. When a GPS is first turned on it will not immediately return any information, it needs certain time to warm up. The warm up time varies by device and can typically take one minute, or even longer if your device lacks line of sight. If device has Internet connection then time to first fix is in about 15-30 seconds. Provider usually allows you to can get information about time-stamp, latitude, longitude, altitude, bearing, speed and accuracy.
- *Real-time Network*: the provider collects IDs from cell networks and WiFi hot spots in the area, then sends this over the Internet to remote Google server which calculates based on this information an approximate location. This is not available in all countries and only works when device has an active Internet connection. Depending on carrier, you can get access to time-stamp, latitude, longitude, altitude, accuracy and speed.

In general, there are several alternatives for obtaining location information [3]:

- *Passive*: the idea is to save battery power, application is listening for new locations which were requested by another application. You may be able to access time-stamp, latitude, longitude, altitude and accuracy, but you will not be able to determine from which provider (GPS or Network) this information was derived.
- *GPS (NMEA*[1] *data)*: Android supports an access to the raw NMEA strings. NMEA format includes complete PVT (position, velocity, time) information gathered by the GPS receiver. This is represented as raw string called sentence, that is totally self contained and independent from other sentence. There are several types of sentences, for example Almanac data (ALM), Fix information (GGA) or Lat/Lon position (GGA).
- *Sensors*: Android smart phones usually have different kinds of sensors such as accelerometer or magnetometer.

Sapiezynski et al. [4] discuss the opportunities and challenges in building WiFi hot spots location database with over 800 mobile devices. They found out that mobility of access points was a very important indicator of accuracy when determining location of device using WiFi hot spots in area. Given their conclusions, we take advantage of WiFi based location acquisition as a trigger for its GPS counterpart.

Currently, a number of free/paid WiFi hot spot and cell tower maps [5], [6], [7] is available. They are based on user location data that were collected when uploading GPS, WiFi hot spot and cell tower locations. These projects can be used to get approximate location by providing ID of cell tower or MAC address of WiFi hot spot nearby.

1. National Marine Electronics Association

Martin et al. [8] developed an application for indoor localization with use of WiFi fingerprinting and accelerometer with accuracy up to 1.5 meters. They showed that accelerometer values are generally too noisy to be considered a reliable source for indoor navigation. For this reason, we focus herein on location acquisition applying only GPS and WiFi.

Kjaergaard et al. [9] proposed an approach for user tracking that takes advantage of built-in accelerometer. However, the current Android API 23 does not allow for accelerometer sensing, when mobile device is in sleep mode. For this reason, their approach was not applicable in our case.

Paek et al. [10] applied accelerometer and Bluetooth for detecting user movement. They also applied cell tower information for deciding whether user is in building, so that GPS location acquisition can be avoided. Similar as in the previous case, since the current Android API does not allow accelerometer operation and cell tower information reading in sleep mode, we had to resort to other options for user movement detection. Additionally, nowadays users often keep Bluetooth interface switched off. Similarly to the previous case, the approach of Youssef et al. [11] cannot be be applied in our case due to Android sleep mode restrictions. The authors used accelerometer and magnetometer readings.

Constandache at al. [12] compared approaches based on GPS, GPS/WiFi, GPS/GSM and GPS/WiFi/GSM, what can be mapped to current Android location providers. Their architecture was built upon a Nokia mobile with Symbian OS. Nowadays, these providers are part of Android API. Their approach takes advantage of user trajectory prediction based on user location history that allows for a better control user location acquisition frequency, thus resulting in improved energy efficiency.

Although, the above discussed related work gives answers on how to obtain location information, in our view, it does not propose satisfactory solutions to the problem of continuous long-term user location tracking. This is especially necessary when users are engaged in A/B, usability or Quality of experience tests that require that location tracking is energy efficient and transparent to the user.

3. Problem Formulation

In this section we introduce the data model used in this paper and the problem formulation.

3.1. Data Model

When a device is turned on and has location services enabled, then we can efficiently record time-stamped locations. We define the simplified location record as follows:

Definition 1. A *location* is a tuple $L = (provider, latitude, longitude, accuracy, timeStamp, BSSID, wifiScanTimeStamp)$,

where $provider \in \{GPS, network\}$. $accuracy$ is the location accuracy with 68% confidence, more precisely we

draw a circle at the center of latitude and longitude of a given location, this circle represented this way has the probability of 68% that it contains the exact location of the device. $timeStamp$ is the Unix epoch time in milliseconds, $BSSID$ is basic service set identifier [13] and $wifiScanTimeStamp$ is the time of the last WiFi scan. If $BSSID$ or $wifiScanTimeStamp$ is unavailable then NULL is inserted. If device was connected to WiFi network when retrieving location, then location record contains unique identifier, $BSSID$ of this WiFi network. We formally define a wifi scan as follows:

Definition 2. A *wifiScan* is a sequence of pairs $S = (t_1, W_1), (t_2, W_2), ..., (t_n, W_n)$,

where for $i = 1...n$, t_i is a non-decreasing time variable that corresponds to $wifiScanTimeStamp$ and W_i denotes a WiFi network W_i. WifiScan is thus a set of WiFi networks in range of device radio at time given by t_i. WiFi network W_i is in turn defined as follows:

Definition 3. A *WiFi network* is a tuple $W = (BSSID, SSID, frequency, level, tsf, capabilities)$,

where $SSID$ is service set identifier (also known as network name) [13], $frequency$ is the frequency in MHz over which device is communicating with access point, $level$ is signal strength in dBm, tsf is time in microseconds for Timing Synchronization Function (TSF) [14] specified in IEEE 802.11 [15] and $capabilities$ records the authentication, key management, and encryption schemes supported by the access point.

3.2. Problem Characterization

Our aim is continuous user location tracking using mobile devices equipped with GPS receiver, however, GPS location acquisition is conditional upon the outcome of WiFi scanning. The purpose of WiFi scanning is to detect whether the user has changed its location significantly.

In our efforts we rely on various location providers as discussed above. An obvious choice is the GPS provider that can deliver location accuracy of up to 10 meters. GPS looks like an ideal location provider, however, GPS also has its draw-backs. The initial connection to satellites, TTFF, can be quite slow. GPS provider needs to connect to at least three satellites for a 2D location (without altitude, which is enough for us), and initial transmission of so-called Ephemeris data takes a minimum of 30 seconds. Once the Ephemeris data has been transmitted and connection to the satellite is established, location updates can be as frequent as we demand. If device has an active Internet connection, then A-GPS can be applied to speed up TTFF to 5-15 seconds. Since Internet based Ephemeris information download is much faster and A-GPS can completely bypass the GPS chip, it also leads to lower power consumption. Nevertheless, eventually GPS starts operating and this can lead to a fast battery drain.

Beside the GPS provider the network location provider uses both WiFi hot spots and cell towers in range of the Android device in order to approximate user location. When the location provider is polled, the IDs of the WiFi hot spots and cell towers in the area are sent via Internet to the Google Location Server, a database with location information on WiFi hot spots and cell towers. Google Location Server then returns an approximate location of the user.

The data comes from a combination of cell and WiFi hot spot information, and can only be obtained when the device can access the Internet to query the Google Location Server. Accuracy of the data is specified somewhere between 100–1,000 m depending on the information available. WiFi hot spots allow an accuracy of 100–500 m, while cell towers only allow for an accuracy greater than 500 m. The result is that the network provider will be very inaccurate in areas without WiFi hot spots. Conversely, the more WiFi hot spots and cell towers are available in the area the greater the accuracy is.

The network provider can give first location within seconds. Compared to GPS location provider, network provider is much faster. If the user disables WiFi, the network provider can not longer use the WiFi receiver and will be limited only to cellular towers to determine approximate location. Cellular towers generally provide very limited accuracy (not accurate enough for the purpose of our research).

If we register network provider to listen for locations, but device is in Airplane mode, network location will never occur. Network provider does not distinguish between not moving and seemingly not moving because of Airplane mode.

There are three concerns of balancing between optimal accuracy, power consumption when acquiring a location:

1) *Multiple location providers* – There is more than one provider from which location can get acquired (GPS, WiFi and Cell-ID) and they all vary in power consumption and accuracy.
2) *User movement* – Location data must get refreshed on a reasonable time interval or be stopped depending on user movement.
3) *Varying accuracy* – Location information coming from each provider may have varying accuracy, even if obtained from the same location provider.

Each one from these three concerns must be considered in order to obtain a suitable trade-off for continuous location recording. The properties of each available location provider are summarized in Table 1. Given the discussion above, we can formulate the problem statement investigated herein as follows:

An *energy efficient device location tracker* should run continuously in background, so that it records each reasonably large user movement, it must not significantly influence battery consumption and its operation must be transparent to the user.

TABLE 1. ACCURACY-BATTERY-TECHNOLOGY OVERVIEW.

Accuracy	Power Usage	Technology
10m	High	*Autonomous GPS* *Provider: GPS* **1.** uses GPS chip on the device **2.** line of sight to GPS satellites necessary **3.** needs min. 3 to get a 2D fix **4.** takes a long time to get a fix **5.** does not work around tall buildings
100m	Medium – Low	*Assisted GPS (A-GPS)* *Provider: network* **1.** uses GPS chip on device, as well as assistance from the network (cellular network) to provide a fast initial fix **2.** medium power consumption **3.** very accurate **4.** works without any line of sight to the sky **5.** depends on carrier and phone supporting this (even if phone supports it, and network does not then this does not work)
1 km	Low	*Cell-ID/WiFi look-up* *Provider: network or passive* **1.** very fast lock, and does not require GPS chip on device to be active, **2.** requires no extra power at all **3.** has very low accuracy

4. Experimental Results

4.1. Android Implementation Details

In this section we first discuss how to obtain location updates and then introduce our method based on WiFi scanning.

When an Android device is left idle, the screen will first dim, then turn off, and then CPU will be ultimately turned off (deep sleep). This prevents the device's battery from quickly getting drained. However, there are times when an application needs to wake up the screen or the CPU and keep it awake in order to complete some computation. To summarize, we define sleep, awake and wake up state as follow:

- *Deep sleep* is a state when device has CPU or screen turned off.
- *Awake* is a state when at least CPU is running.
- *Wake up* is a state when device comes from deep sleep to awake state.

Registering for obtaining location updates can be done in two different ways:

- *Active way*, when you register listener, which is listening for possible location updates. This way, device must stays awake, what results in *high power usage*.
- *Passive way*, using a wakeful broadcast receiver and intents. This way allows device to be in deep sleep until location update occurs. When wakeful broadcast receiver receives location update via Android intent, we can decide to wake up and keep device awake during location processing if needed. This option is favourable to active way, since it results in *low power usage*.

The location update interval can be controlled in both ways by Android API using the $minTime$ and $minDistance$ parameter. Then the elapsed time between two location updates will never be less than $minTime$.

If the $minDistance$ parameter is greater than 0, then the location provider will only send an update when distance of the location has changed by at least $minDistance$ meters and at least $minTime$ milliseconds have passed. With using parameter $minDistance$ location providers can not save much energy because location was already gathered to calculate distance. Therefore in order to save battery life, $minTime$ should be the primary tool.

We tested both ways of recording locations updates. But any of these implementations did not fulfil our problem statement. Using GPS provider in deep sleep was consuming excessive battery power and such an implementation failed to remain transparent to the user. Needless to say, this implementation also resulted in high power usage when GPS was trying to obtain fix in buildings. For these reasons, we decided to focus on cascading detection, where WiFi scanning is used as a trigger for location acquisition with GPS. In the following, we explain how we implemented WiFi scanning.

Scanning for WiFi hot spots in range can be done in Android in two ways:

- *Active way*, a device needs to tune in its radio to a particular channel and then it transmits probe request. After this it waits about 50 milliseconds for probe responses from access points on that channel. This process is repeated by device over all accessible channels. This procedure is faster compared to passive scan, but it also consumes more energy, since the radio also transmits frames, not only receives them. Device must stay awake between initializing scan and obtaining response, what results in *high power usage*.
- *Passive way* is slower to perform, because the device needs to listen on every channel for some period of time, waiting for broadcast beacons. Beacon frames are transmitted by access points periodically to announce the presence of a wireless LAN. Transmitted beacon frames contain complete information about its network. This approach consumes *less energy*, since the radio does not use transceiver, but only the receiver. It also takes more time to finish, since it has to listen on every channel for a certain period of time. Device can be in deep sleep during passive scan.

According to our tests, devices with Android use passive scans for gathering info about surrounding WiFi hot spots.

TABLE 2. TIME INTERVAL BETWEEN SUCCESSIVE SCANS IN SECONDS.

Device model	Average	Max.	Min.	Median
Lenovo X2 Vibe	95	134	3	122
Lenovo Tab S8	110	183	5	115
Lenovo Yoga 2	98	129	4	108

As we can see in Table 2, the interval between successive scans varies significantly with respect to device used. In order to keep energy consumption as low as possible, in our approach we decided to take advantage of WiFi information acquired only through passive scans, i.e. *we do not initiate any WiFi scans*.

In addition to the above said, another research challenge for reasonable energy-accuracy trade-off during continuous location recording is to detect when location provider changes and based on that adaptively pause sensing for better power consumption. For solving this problem we use passive way of recording location and WiFi scans, and formulate these cases of interest:

- Device is *connected to WiFi* access point (AP) with Internet connection.
- Device is in *coverage of static WiFi* AP, but is not connected to any WiFi hot spot,
- Device in range of a *moving WiFi* AP, connected or not to to this moving WiFi hot spot. A moving hot spot is a mobile device installed in bus, train, airplane etc.
- Device is in area with *no WiFi* AP.
- Device is having its *WiFi radio turned off*.

The first case can be implemented with Android WakefulBroadcastReceiver listening for action "CONNECTIVITY CHANGE". If connection to WiFi AP with Internet access is detected, then we replace current provider with network and after first retrieved location we stop any provider sensing to save battery life.

For scanning WiFi AP change, we decided to use passive scans (default for Android, so Android can manage timing itself) for lowest power usage. We will only listen with WakefulBroadcastReceiver for actions "SCAN RESULTS" on WiFi APs change.

If APs around device do not change, then we believe that device is not moving. Based on this assumption, we stop any provider scanning until WiFi APs change. We call this case *static* WiFi APs. The details are shown in Algorithm 1.

In Algorithm 1 we apply Timing Synchronization Function (TSF) [14] to eliminate temporary WiFi APs created by users with smart-phones. In our experiments we set $time = 5$ hours and $\gamma = 0.6$. We believe that WiFi hot spots operating longer than 5 hours, are stable for localization. The values for both parameters were established experimentally.

When we are connected to WiFi network or only in area of static WiFi networks, then we turn off location sensing. But what if that WiFi network is moving, for example it is a train/bus WiFi AP? Then our approach returns false positive. In order to solve this problem we came up with detection of

Data: previous and current $wifiScan$: S_{prev}, S_{now}
Result: Boolean: presence in static WiFi APs
if S_{now} *is empty OR* S_{prev} *is empty* **then**
  return false;
end
same := 0;
count := 0;
forall the $W_i \in S_{now}$ **do**
  if $W_i(TSF) < time$ **then**
    continue;
  end
  if $W_i \in S_{prev}$ **then**
    same++;
  end
  count++;
end
if $count > 0$ *AND* $\frac{same}{count} < \gamma$ **then**
  return true;
end
return false;

Algorithm 1: Check static WiFi hot spots

such moving WiFi APs. When moving WiFi AP is detected then GPS provider must start sensing for locations.

In order to detect moving WiFi AP we must first collect at least k instances of $wifiScan$. What we implement is a sliding window approach, where we evaluate the change in WiFi APs in k instances of $wifiScan$. More formally, we proceed as follows:

$$M = \bigcap_k g(wifiScan_k),$$

where M is the set of distinct networks present in k instances of $wifiScan$ and g is a projection function:

$$g : wifiScan_k \rightarrow \{BSSID_i\}.$$

If $M \neq \emptyset$ then we compute:

$$MM = \biguplus_k g(wifiScan_k),$$

where MM is the multiset of networks present in k instances of $wifiScan$. The element frequency (replication) function f is then defined as:

$$f : MM \rightarrow \mathbb{N},$$

where $\mathbb{N}$ is the set of natural numbers. Then we compute for each $x \in MM$ such that $x \notin M$, the relative element frequency as:

$$\alpha_x = \frac{f(x)}{\sum_{y \in MM} f(y)}.$$

We only consider WiFi APs W_i with $\alpha_x < \alpha_0$. The rationale of this approach is to verify whether (i) there is a WiFi AP that remains present in all k instances of $wifiScan$, and (ii) the remaining WiFi APs change frequently.

In our experiments, we set $k = 10$ and $\alpha_0 = 0.5$. As in the case of $time$ and γ, the values for these parameters were established experimentally.

TABLE 3. TIME TO FULLY DISCHARGE COMPARED.

Method	Lenovo X2 Vibe	Lenovo Tab S8
Active GPS	5 hours	6 hours
Passive GPS	9 hours	11 hours
Our approach	26 hours	29 hours

TABLE 4. LOCATIONS UNTIL ANY MOBILE HAS DISCHARGED.

Method	Lenovo X2 Vibe	Lenovo Tab S8
Active GPS	16,231 locations	16,894 locations
Passive GPS	15,792 locations	16,324 locations
Our approach	2,874 locations	3,566 locations

4.2. Results

In order to offer a fair comparison of our approach to other approaches, we considered two baseline cases:

- *Active GPS*: this base case uses GPS listener at all times. Device thus cannot enter sleep state.
- *Passive GPS*: this base case uses WakefulBroadcastReceiver to listen for actions "LOCATION CHANGED" for retrieving locations and never turned off. Device during this method was repeatedly waking up from deep sleep when location change occurred.

In Tables 3 and 4 we show a comparison of our approach with two baseline cases realized on two different kinds of mobile devices, which were carried by the same person. This allowed for a direct comparison, since the real user trajectory was identical. The tables show the time necessary for a device to reach a completely discharged state and the number of collected locations.

In order to give the reader basic idea on the performance of our approach with respect to the passive GPS approach, we include Figures 1 and 2. It can be seen that our approach leads to less locations compared to both active and passive GPS location recording. However, our approach still allows for understanding the substantial user movement, what was our goal.

5. Conclusions and Future Work

We presented an approach which adaptively starts/stops GPS sensing based on WiFi scanning in order to obtain a suitable trade-off between location precision and battery life. The proposed method for location recording significantly decreases battery consumption based on using multiple techniques that adapt device to current situation.

We recently acquired 1,080 mobile devices that we distributed among our students. We are working on a comprehensive test that requires energy efficient location tracking, that can be applied, for example, in user trajectory prediction. In the future, we also plan to take advantage of various available WiFi and cell tower maps [5], [6], [7].

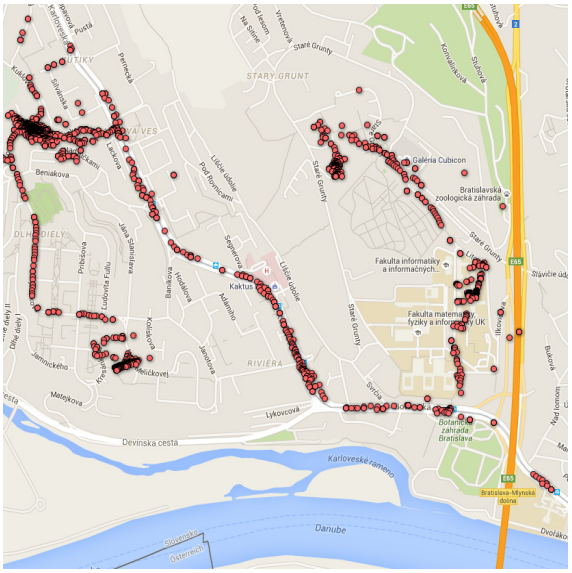

Figure 1. Location information collected by Passive GPS method: 8,127 locations.

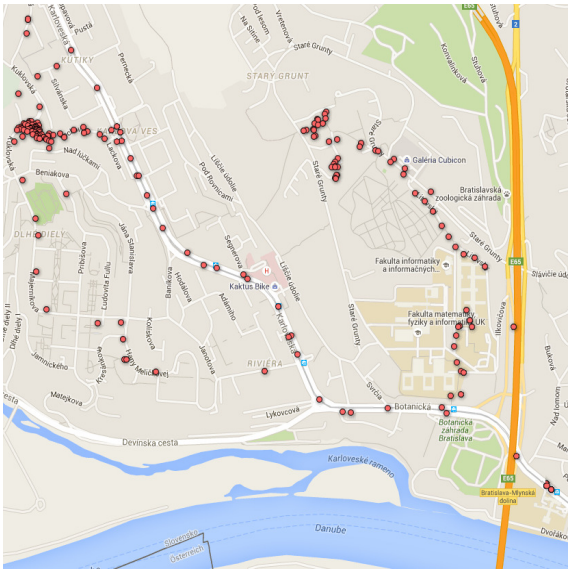

Figure 2. Location information collected by our approach: 195 locations.

References

[1] S. Marcek and M. Drozda, *Proceedings of the Mediterranean Conference on Information & Communication Technologies 2015: MedCT 2015 Volume 2*. Cham: Springer International Publishing, 2016, ch. Predicting System Failures on Mobile Devices, pp. 499–508. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-30298-0_51

[2] M. Rokicki and M. Drozda, "Evaluating trade-offs in energy-efficient error detection," *International Journal of Communication Systems*, 2015. [Online]. Available: http://dx.doi.org/10.1002/dac.3028

[3] Google, "LocationManager — Android Developers," 2016, Accessed: 19-May-2016. [Online].

Available: http://developer.android.com/reference/android/location/LocationManager.html

[4] P. Sapiezynski, R. Gatej, A. Mislove, and S. Lehmann, "Opportunities and challenges in crowdsourced wardriving," in *Proceedings of the 2015 ACM Conference on Internet Measurement Conference*. ACM, 2015, pp. 267–273.

[5] Skyhook Wireless, Inc., "Skyhook — global location and context software products," 2016, Accessed: 19-May-2016. [Online]. Available: http://www.skyhookwireless.com/

[6] "The OpenCellID map," 2016, Accessed: 19-May-2016. [Online]. Available: http://opencellid.org/

[7] Combain Mobile AB, "Wifi positioning — wifi location — cell id - combain," 2016, Accessed: 19-May-2016. [Online]. Available: https://combain.com/

[8] E. Martin, O. Vinyals, G. Friedland, and R. Bajcsy, "Precise indoor localization using smart phones," in *Proceedings of the international conference on Multimedia*. ACM, 2010, pp. 787–790.

[9] M. B. Kjærgaard, J. Langdal, T. Godsk, and T. Toftkjær, "Entracked: energy-efficient robust position tracking for mobile devices," in *Proceedings of the 7th international conference on Mobile systems, applications, and services*. ACM, 2009, pp. 221–234.

[10] J. Paek, J. Kim, and R. Govindan, "Energy-efficient rate-adaptive gps-based positioning for smartphones," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*. ACM, 2010, pp. 299–314.

[11] M. Youssef, M. A. Yosef, and M. El-Derini, "GAC: energy-efficient hybrid GPS-accelerometer-compass GSM localization," in *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*. IEEE, 2010, pp. 1–5.

[12] I. Constandache, S. Gaonkar, M. Sayler, R. R. Choudhury, and L. Cox, "Enloc: Energy-efficient localization for mobile phones," in *INFOCOM 2009, IEEE*. IEEE, 2009, pp. 2716–2720.

[13] "IEEE standard for information technology - telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications," *IEEE Std 802.11-2007 (Revision of IEEE Std 802.11-1999)*, pp. 1–1076, June 2007.

[14] J. Lee, R. Etkin, S. Lee, A. Gupta, and V. Ma, "Synchronization in a wireless node," Sep. 8 2011, US Patent App. 12/715,774. [Online]. Available: http://www.google.com/patents/US20110216660

[15] "Information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications," *ISO/IEC/IEEE 8802-11:2012(E) (Revision of ISO/IEC/IEEE 8802-11-2005 and Amendments)*, pp. 1–2798, Nov 2012.